Universidad
Carlos III de Madrid
www.uc3m.es

Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Telemática

Servicio Web de telefonía SIP

Trabajo Fin de Grado

Autor: Sergio Crespo Bellido

Tutor: Jesús Arias Fisteus

Director en empresa: Manuel Nuñez Sanz

Marzo 2014

Universidad Carlos III de Madrid
www.uc3m.es

Trabajo Fin de Grado

Servicio Web de telefonía SIP.

Autor
Sergio Crespo Bellido

Tutor
Jesús Arias Fisteus

Director en empresa
Manuel Nuñez Sanz

Lectura del Trabajo Fin de Grado el día 4 de Marzo de 2013 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, el tribunal:

PRESIDENTE: **Alberto García Martínez**

SECRETARIO: **José Joaquín Escudero Garzas**

VOCAL: **Alberto Cortes Martín**

Leganés, 4 de Marzo de 2014

Resumen

Este trabajo, desarrollado en Telefónica, consiste en el desarrollo de una aplicación web en Flash que permita la comunicación entre usuarios SIP, números fijos y móviles. La aplicación permitirá al usuario, emulando el aspecto de un terminal móvil, realizar y recibir llamadas de audio y vídeo, así como enviar mensajes de texto (SMS) y tener un registro de las llamadas.

En segundo lugar se realiza un estudio de nuevas tecnologías web como WebRTC. Con esta tecnología se desarrolla en HTML5 y JavaScript una nueva aplicación que ofrece mayor compatibilidad de dispositivos y no requiere instalación de plugins adicionales.

En esta memoria se explican los protocolos y tecnologías utilizadas y se detallan las partes más significativas en el desarrollo de ambas aplicaciones.

Índice General

Resumen v

Índice General vi

Capítulo 1: Introducción. 1

1.1 Motivación del proyecto. 1

1.2 Objetivos. 1

1.3 Plan de trabajo. 2

1.4 Presentación del resto de memoria. 5

Capítulo 2: Estado del Arte 6

2.1 SIP. 6

2.2 RTMP. 8

2.2 IMS. 10

OpenIMSCore. 11

2.3 Flash. 11

Red5 Server. 12

Red5phone. 12

2.4 HTML5 y CSS3. 14

HTML5 en dispositivos móviles. 16

2.5 WebRTC 16

WebRTC2Sip. 20

Sipml5. 21

Capítulo 3: Requisitos 22

Capítulo 4: Arquitectura del sistema 24

Capítulo 5: Servicio con cliente Flash. 26

Capítulo 6: Servicio con cliente HTML5 y WebRTC. 35

Cliente para dispositivos móviles en HTML5: 39

Capítulo 7: Recepción, procesado y gestión de datos. 41

7.1 Archivos XML 41

7.2 Fichero JSON 45

7.3 PHP 46
7.4 Gestión 49
Capítulo 8: Plan de pruebas. 52
8.1 Versión Flash 52
8.2 Versión HTML5. 54
8.3 Versión Móvil. 55
Capítulo 9: Conclusiones y trabajos futuros 56
Presupuesto. 57
Marco regulador 60
Entorno socio-económico. 61

Lista de figuras 62
Lista de tablas 62
Acrónimos 63
Referencias 66

Capítulo 1: Introducción.

Este documento presenta el Trabajo de Fin de Grado basado en el desarrollo de una aplicación web desarrollada en Flash que permite al usuario realizar llamadas de audio y vídeo a través de internet. Por otro lado consiste en el estudio para el desarrollo de la misma aplicación utilizando HTML5 y WebRTC como alternativa.

En este primer capítulo se presentan las motivaciones que han llevado al desarrollo del proyecto, los objetivos que se han querido lograr y el plan de trabajo acordado.

1.1 Motivación del proyecto.

Las telecomunicaciones han sufrido grandes avances tecnológicos en los últimos años. Nada tiene que ver el teléfono inventado por Alexander Graham Bell con la infraestructura desplegada en la actualidad. Las redes de conmutación manuales permitían el establecimiento de una llamada gracias a la intervención humana. Ya en el siglo XX aparecen las centrales automáticas, siendo el establecimiento de manera mecánica. Estos sistemas analógicos dieron paso a los sistemas digitales. La voz es digitalizada así como la señalización permitiendo ofrecer diferentes servicios. [1] En la actualizad con la línea de ADSL o el uso de la fibra óptica el usuario dispone no solo de una línea con la que llamar sino de la posibilidad de comunicarse a través de internet.

El uso de internet ofrece servicios y aplicaciones que permiten al usuario comunicarse en todo momento con amigos, familiares o conocidos, ya sea enviando mensajes de texto, imágenes o vídeos, compartiendo experiencias en las redes sociales, etc.

Este trabajo muestra una aplicación desarrollada en Telefónica I+D, que permite la comunicación únicamente con tener conexión a internet y un navegador web. Independientemente del dispositivo desde el cual se conecte a la aplicación web dispondrá siempre del mismo número de contacto.

1.2 Objetivos.

El principal objetivo es ofrecer una aplicación web desarrollada en Flash que permita al usuario realizar llamadas a números móviles, fijos o a otros usuarios registrados en la aplicación. Además, deberá ofrecer otros servicios como poder recibir llamadas, enviar SMSs, realizar videollamadas entre usuarios de la aplicación y disponer de un registro donde consultar las llamadas realizadas o recibidas.

El segundo objetivo de este proyecto es el estudio del desarrollo de una alternativa a esta primera aplicación web utilizando las tecnologías HTML5 y WebRTC. Deberá ofrecer características similares a la anterior además de poder ser utilizada en terminales móviles, por lo que se implementará una interfaz adecuada a estos dispositivos.

Por último, ambas aplicaciones deberán generar registros de su utilización para poder obtener estadísticos. Para ello, se diseñará una web que muestren estos registros de manera ordenada y legible y que además permitirán al administrador gestionar los usuarios y contraseñas de acceso.

1.3 Plan de trabajo.

En esta sección se muestran las diferentes tareas llevadas a cabo durante la realización del proyecto. Se ha separado en dos bloques principales, Proyecto y Memoria. El proyecto a su vez en otros dos, diferenciando la aplicación Flash de la de HTML5.

A continuación se muestran organizadas en diferentes apartados según la etapa de desarrollo.

Proyecto

1.1. Aplicación Flash

Documentación:

Se estudiará el servidor web (Apache HTTP), los protocolos utilizados (SIP y RTMP), y los diferentes sistemas y programas (IMS, Red5Server, Red5phone y OpenIMSCore).

Instalación y configuración:

Instalación del S.O (Ubuntu) en los equipos, servidor HTTP, acceso remoto para la gestión y otras aplicaciones necesarias. Instalación del servidor Flash, configurando y compilando el código fuente. Configuración OpenIMSCore (acceso e introducción de los usuarios). Instalación de la versión original Red5phone y prueba de su funcionamiento.

Desarrollo:

Cambios del código fuente del servidor Red5Server, compilación y ejecución. Desarrollo e implementación de aplicación Red5phone modificando el código inicial añadiendo funcionalidades y modificando el diseño. Desarrollo de funcionalidades con PHP y lectura, escritura y borrado de ficheros XML.

Pruebas y registros de gestión:

Pruebas de las funcionalidades en los diferentes navegadores (acceso, llamada, vídeo, envío de SMS, etc). Diseño de una web con registro de las llamadas realizadas por los usuarios para comprobar si el funcionamiento es correcto. Consulta de las encuestas enviadas por los usuarios para observar las valoraciones de los usuarios. Registro de toda la información recopilada y

volcado de los datos a fichero en formato CSV para su estudio estadístico y por último gestión de los usuarios mediante configuración de usuarios y contraseñas desde la web.

1.2. Aplicación HTML5

Documentación:

Estudio de los lenguajes utilizados (HTML5, CSS3 y JavaScript) y documentación sobre la tecnología WebRTC.

Instalación y configuración:

Instalación de los componentes para el desarrollo y correcto funcionamiento de la aplicación en HTML5, Webrtc2sip y Sipml5.

Desarrollo:

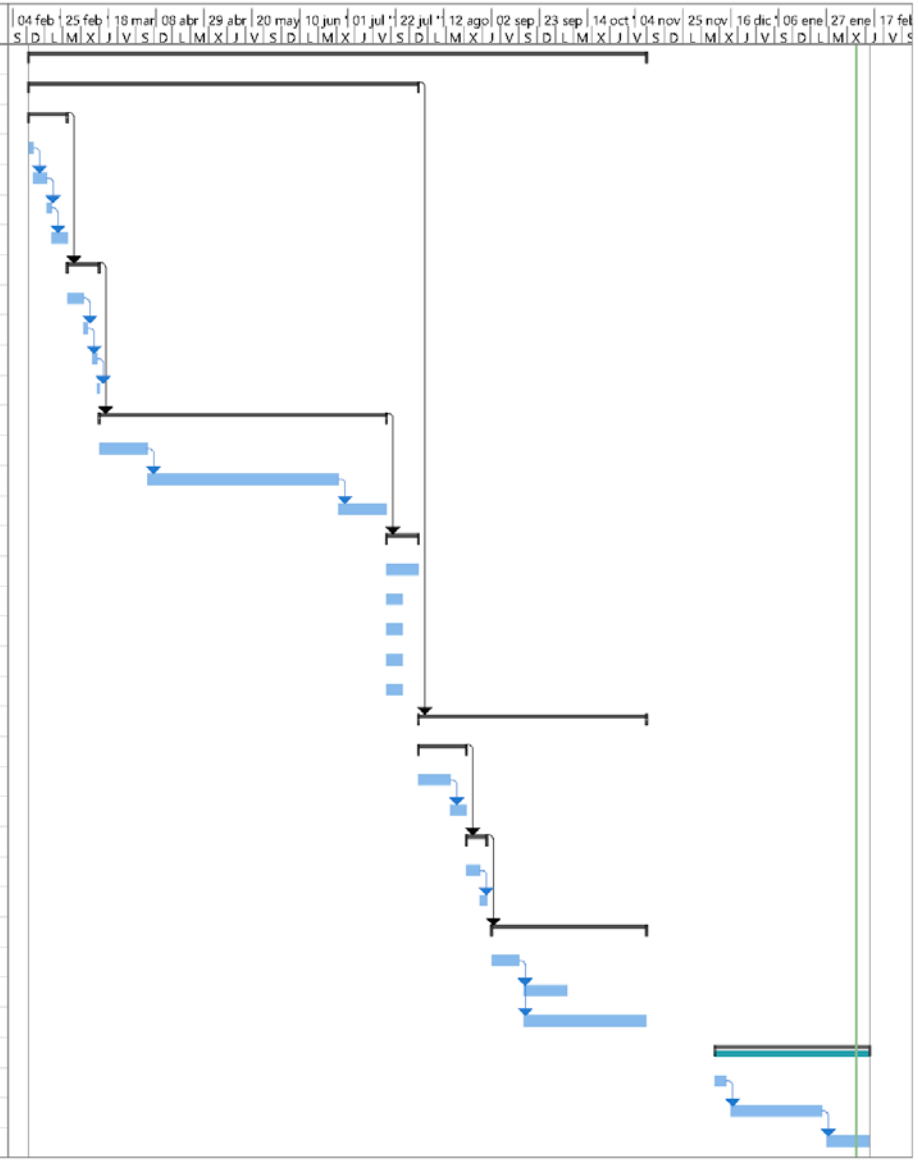
Modificaciones y reestructuración del código Sipml5, diseño de la aplicación acorde a la versión Flash e implementación de las funcionalidades añadidas.

2. Memoria

Planificación y estructura
Redacción
Correcciones

En la figura 1 se muestra el diagrama de Gantt. Se debe tener en cuenta que las fechas y duraciones de las tareas son orientativas. El desarrollo de este trabajo se ha realizado durante el periodo lectivo universitario.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**1 Proyecto**	**195 días**	**lun 11/02/13**	**vie 08/11/13**
2	**1.1 Aplicación Flash**	**123 días**	**lun 11/02/13**	**mié 31/07/13**
3	**1.1.1 Documentación**	**13 días**	**lun 11/02/13**	**mié 27/02/13**
4	1.1.1.1 Apache HTTP	2 días	lun 11/02/13	mar 12/02/13
5	1.1.1.2 Protocolo SIP y RTMP	4 días	mié 13/02/13	lun 18/02/13
6	1.1.1.3 IMS	2 días	mar 19/02/13	mié 20/02/13
7	1.1.1.4 Red5Server, red5phone y OpenIMSCore	5 días	jue 21/02/13	mié 27/02/13
8	**1.1.2 Instalación y configuración**	**10 días**	**jue 28/02/13**	**mié 13/03/13**
9	1.1.2.1 Equipos	5 días	jue 28/02/13	mié 06/03/13
10	1.1.2.2 Red5Server	2 días	jue 07/03/13	vie 08/03/13
11	1.1.2.3 IMSCore	2 días	lun 11/03/13	mar 12/03/13
12	1.1.2.4 red5phone	1 día	mié 13/03/13	mié 13/03/13
13	**1.1.3 Desarrollo**	**90 días**	**jue 14/03/13**	**mié 17/07/13**
14	1.1.3.1 Modificaciones Red5Server	15 días	jue 14/03/13	mié 03/04/13
15	1.1.3.2 Implementación aplicación flash	60 días	jue 04/04/13	mié 26/06/13
16	1.1.3.3 Desarrollo funcionalidades PHP	15 días	jue 27/06/13	mié 17/07/13
17	**1.1.4 Pruebas registros y gestión**	**10 días**	**jue 18/07/13**	**mié 31/07/13**
18	1.1.4.1 Pruebas flash	10 días	jue 18/07/13	mié 31/07/13
19	1.1.4.2 Desarrollo web registros	5 días	jue 18/07/13	mié 24/07/13
20	1.1.4.3 Desarrollo web encuestas	5 días	jue 18/07/13	mié 24/07/13
21	1.1.4.4 Desarrollo web gestión de usuarios	5 días	jue 18/07/13	mié 24/07/13
22	1.1.4.5 Desarrollo web estadisticas	5 días	jue 18/07/13	mié 24/07/13
23	**1.2 Aplicación HTML5**	**72 días**	**jue 01/08/13**	**vie 08/11/13**
24	**1.2.1 Documentación**	**15 días**	**jue 01/08/13**	**mié 21/08/13**
25	1.2.1.1 HTML5, CSS3 y JavaScript	10 días	jue 01/08/13	mié 14/08/13
26	1.2.1.2 WebRTC	5 días	jue 15/08/13	mié 21/08/13
27	**1.2.2 Instalación y configuración**	**7 días**	**jue 22/08/13**	**vie 30/08/13**
28	1.2.2.1 Webrtc2SIP	4 días	jue 22/08/13	mar 27/08/13
29	1.2.2.2 sipml5	3 días	mié 28/08/13	vie 30/08/13
30	**1.2.3 Desarrollo**	**50 días**	**lun 02/09/13**	**vie 08/11/13**
31	1.2.3.1 Modificación sipml5	10 días	lun 02/09/13	vie 13/09/13
32	1.2.3.2 Diseño aplicación	15 días	lun 16/09/13	vie 04/10/13
33	1.2.3.3 Implementación funciones añadidas	40 días	lun 16/09/13	vie 08/11/13
34	**2 Memoria**	**50 días**	**lun 09/12/13**	**vie 14/02/14**
35	2.1 Planificacion y estructura memoria	5 días	lun 09/12/13	vie 13/12/13
36	2.2 Redaccion de la memoria	30 días	lun 16/12/13	vie 24/01/14
37	2.3 Correcciones de la memoria	15 días	lun 27/01/14	vie 14/02/14



1.4 Presentación del resto de memoria.

En esta sección se describe de manera resumida cada uno de los Capítulos de esta memoria.

En el capítulo 2 se estudian los protocolos y tecnologías más importantes en el desarrollo de este trabajo. Se explicará el protocolo SIP, utilizado como mecanismo para establecer llamadas a través de una red IP, el protocolo RTMP utilizado para la transmisión entre el servidor Flash y el cliente desarrollado e IMS como arquitectura de servicios multimedia. Se introduce Flash, el servidor Red5 y la aplicación Red5phone. Se estudian las nuevas novedades que nos ofrecen HTML5 y CSS3 con especial énfasis en WebRTC, que permite realizar una comunicación de audio, vídeo o datos entre navegadores sin necesidad de Flash u otros complementos y por último introduce el Gateway WebRTC2SIP y el cliente HTML, Sipml5.

En el capítulo 3 se listan los requisitos que deben cumplir las aplicaciones desarrolladas detallando cada uno de ellos.

El capítulo 4 describe la arquitectura, los equipos necesarios en el despliegue y la explicación de su funcionamiento.

El capítulo 5 presenta la aplicación desarrollada en Flash, las decisiones tomadas en el diseño, cómo acceder al micrófono y la cámara, cómo transmitir y recibir los datos con el servidor Flash, así como los diferentes estados de la aplicación y la comunicación con el servidor web.

En el capítulo 6 se explica el desarrollo de una aplicación similar implementada mediante HTML5. Ofrece la misma funcionalidad que la aplicación Flash, pero utiliza WebRTC con la ayuda de la biblioteca de código Sipml5. Además se comentará su adaptación a dispositivos móviles.

El capítulo 7 explica cómo se recogen en el servidor las peticiones realizadas por la aplicación, el formato de los diferentes archivos utilizados para guardar registros y el diseño de la página web donde poder gestionar estos datos.

El capítulo 8 detalla las pruebas realizadas para la aplicación Flash, HTML5 y móvil.

En el capítulo 9 se escriben las conclusiones obtenidas tras el desarrollo del trabajo y se proponen trabajos futuros.

Por último se detalla el presupuesto del trabajo, el marco regulador y el entorno socio-económico.

Capítulo 2: Estado del Arte.

En este segundo capítulo se describirán de manera resumida los diferentes protocolos y tecnologías utilizadas durante la realización del proyecto.

2.1 SIP.

SIP, Session Initiation Protocol, es un protocolo de señalización que proporciona un mecanismo para establecer llamadas a través de una red IP. Funciona en el nivel de aplicación y puede ser utilizado sobre varios protocolos de transporte. Permite establecer y notificar que se desea iniciar una comunicación, acordar entre los participantes los métodos de codificación de los datos multimedia y dar por terminadas las llamadas [2] [3]. Además proporciona mecanismos para determinar la dirección IP del receptor y la gestión de llamadas como añadir nuevos flujos multimedia, cambiar método de codificación o invitar a nuevos participantes [4] [5].

Elementos SIP.

El protocolo SIP define varias entidades:

- Agente de usuario (UA - User Agent): Es la entidad SIP que interactúa con el usuario.
- Servidor Redirector (Redirect Server): Ayudar a localizar a los UA proporcionando lugares alternativos donde el usuario puede ser alcanzable.
- Servidor Proxy (Proxy Server): Encamina las solicitudes o respuestas a su destino. Están diseñados para que sean transparentes al usuario.
- Servidor Registrador (Registrar Server): Gestiona la información de localización de usuarios en un determinado dominio.

Mensajes SIP.

SIP es un protocolo petición-respuesta cuyos mensajes se codifican en texto y no en binario.

Respecto a la los mensajes de petición la especificación básica de SIP define seis tipos de solicitudes SIP, cada uno de ellos con un propósito diferente. Cada petición SIP contiene un campo, los más usados son:

- INVITE Iniciar sesión multimedia.
- ACK Confirmar respuestas definitivas a solicitudes INVITE.
- OPTIONS Solicitar capacidades del UA. (ej. Protocolos, codecs, etc.)
- BYE Terminar sesión multimedia.
- CANCEL Cancelar solicitudes INVITE pendientes.
- REGISTER Registrar a un usuario en una localización de red.

Existen otros métodos SUBSCRIBE, NOTIFY, PUBLISH, REFE, etc.

El otro tipo de mensajes, las respuestas SIP contienen al principio del mensaje lo que se denomina “Status-line” que contiene la versión del protocolo (siempre es “SIP/2.0”), un código de resultado de tres dígitos y una descripción textual del código.

Rango	Tipo respuesta	Ejemplos
100-199	Informativa	100 Trying, 180 Ringing.
200-299	Éxito	200 OK, 202 Accepted.
300-399	Redirección	301 Moved Permanently.
400-499	Error de cliente	400 Bad Request.
500-599	Error de servidor	500 Server Internal Error.
600-699	Fallo global	600 Busy Everywhere.

Formato mensajes SIP: Un mensaje SIP consiste en una línea inicial (línea de petición si es un mensaje de petición o línea de estado si es de respuesta), varias cabeceras, una línea en blanco, y el cuerpo del mensaje. El cuerpo del mensaje es opcional (algunas peticiones o respuestas no lo llevan).

Cabeceras SIP: Las peticiones contienen algunas cabeceras SIP después de la línea de petición, mientras que las respuestas después de la línea de estado. Las cabeceras proporcionan información acerca de la solicitud (o respuesta) y sobre el cuerpo que contiene. Algunas de las cabeceras se pueden utilizar tanto en solicitudes y respuestas, pero otros son específicos. La cabecera consiste en el nombre de cabecera, seguido por dos puntos, seguido por el valor de la cabecera. Algunas de las cabeceras más comunes son:

- Call-ID: Identifica de forma única una solicitud INVITE.
- Contact: Proporciona una URL donde se puede llegar al usuario directamente.
- Cseq: Contiene un número y el nombre del método. Permite ordenar solicitudes, diferenciar entre nuevas solicitudes y retrasmisiones y asociar respuestas con solicitudes.
- To: Especifica el receptor de la solicitud
- From: Indica el iniciador de una solicitud
- Via: Mantiene un registro de los proxies atravesados por una solicitud.

Cuerpo mensajes SIP: Tanto las solicitudes como las respuestas pueden contener cuerpo. Éste se separa de las cabeceras por una línea en blanco. El cuerpo del mensaje transportado por mensajes SIP es por lo general una descripción de la sesión, pero puede consistir en cualquier dato relacionado con la sesión.

2.2 RTMP.

RTMP, Real-Time Messaging Protocol es un protocolo para transmisión de audio, vídeo y datos en streaming utilizando aplicaciones Flash y un servidor web. Fue inicialmente desarrollado por Macromedia aunque en la actualidad pertenece a Adobe.

Implementado sobre TCP/IP, RTMP envía los datos sin cifrar aunque permite la utilización de mecanismos empleando para ello algunas de las variantes a este protocolo: RTMPS, que funciona sobre una conexión TSL/SSL (protocolos criptográficos), RTMPE, que utiliza mecanismos de seguridad de Adobe y RTMPT, que encapsula el tráfico sobre HTTP utilizando un túnel.

RTMP ofrece una serie de ventajas. En la transmisión de vídeo permite al usuario acceder a un momento del vídeo determinado sin tener que descargar la porción de vídeo anterior a ese instante, el contenido descargado se almacena en el buffer de Flash Player y no en el navegador, por lo que ofrece protección a la propiedad intelectual de los contenidos ofrecidos. Además este protocolo requiere una conexión con el servidor por cada transmisión. [6]

Para la transmisión de audio y vídeo en vivo es necesario disponer de un Flash Communication Server (FCS) y un Flash Media Server (FMS). Un ejemplo es Red5, explicado más adelante.

Funcionamiento.

Tras el establecimiento de la conexión TCP (SYN, SYN-ACK, ACK), se inicia la conexión RTMP con el intercambio de seis paquetes. El cliente y el servidor enviarán los mismos paquetes. Los paquetes enviados por el cliente son llamados C0, C1 y C2 mientras que los enviados por el servidor S0, S1 y S2.

S0 y C0 son paquetes de 8bits. C0 identifica la versión de RTMP utilizada y S0 identifica la versión seleccionada por el servidor. Su valor por defecto es 0x03 (versión 3).

S1 y C1 son paquetes de 1536 octetos de longitud. Estos paquetes están formados por los campos Time, Zero y datos aleatorios. Time contiene una marca de tiempo que deberá ser igual en los paquetes futuros, Zero es un campo todo a ceros y los datos aleatorios será una cadena de valores arbitrarios.

S2 y C2 son paquetes de 1536 octetos de longitud y enviados en respuesta a los paquetes S1 y C1. Sus campos son Time, que contiene la misma marca de tiempo que S1 en el caso de S2 o C1 en caso de C2. Time2 contiene las marcas de tiempo de los paquetes S1 o C1 anteriores. Y datos eco, que contiene los datos aleatorios enviados en S1 (para C2) o S2 (para C1).

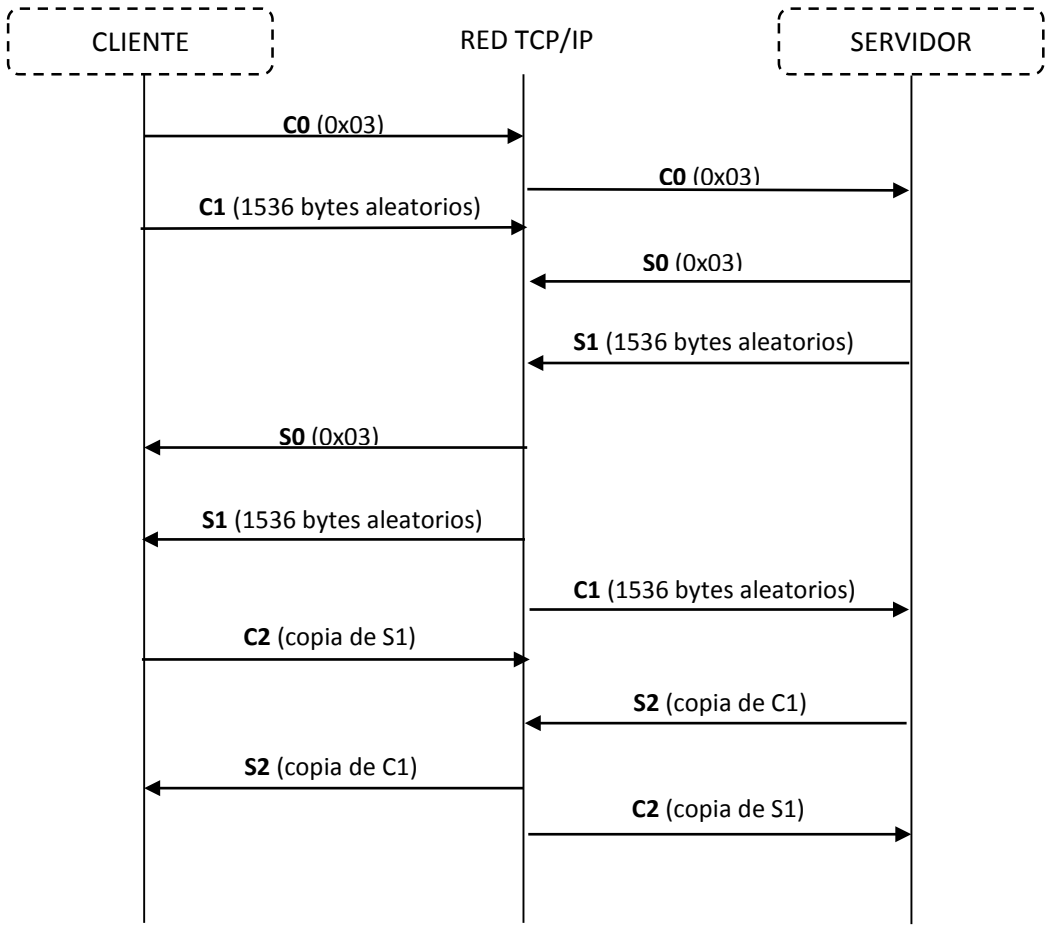


Figura 2: Comunicación cliente-servidor utilizando el protocolo RTMP

RTMP utiliza diferentes mensajes y comandos que permiten al cliente y el servidor comunicarse.

Mensajes:

1. Mensaje de comando:
   Utilizado para realizar operaciones como `connect`, `createStream`, `publish`, `play` o `pause` en el cliente o como `onStatus` o `result` para informar al remitente sobre el estado de los comandos solicitados.
2. Mensaje de datos:
   Utilizado por el cliente o el servidor para enviar metadatos (detalles del audio, vídeo, duración…) o algún dato de usuario al cliente.
3. Mensaje de objetos compartidos:
   Contiene uno o más eventos sobre los objetos compartidos. Como creación, eliminación o modificación.
4. Mensaje de audio:
   Mensaje enviado por el cliente o el servidor para enviar datos de audio.
5. Mensaje de vídeo:
   Mensaje enviado por el cliente o el servidor para enviar datos de vídeo.

6. Mensaje agregado:
   Es un simple mensaje que contiene sub-mensajes RTMP
7. Mensaje de eventos de control de usuario:
   Mensajes enviados por el cliente o el servidor para notificar de los eventos de control de usuario como avisar de que un flujo puede ser utilizado, que no hay más datos en el flujo o la longitud del buffer entre otros.

Comandos:
Estos mensajes están formados por el nombre del comando, un ID y un objeto que contiene parámetros relacionados con el comando. Estos comandos son respondidos con `_result`, `_error` o con el nombre de un método.

Hay dos clases de objetos que permiten enviar varios comandos:

1. NetConnection: Un objeto que es una representación de alto nivel de la conexión entre el servidor y el cliente.
   - *Connect*: Enviado por el cliente al servidor para solicitar una conexión.
   - *Call*: Permite realizar una llamada a un proceso remotamente.
   - *Close*: Para cerrar la conexión.
   - *CreateStream*: Enviado por el cliente al servidor para crear un canal lógico para la comunicación.
2. NetStream: Un objeto que representa el canal sobre el cual se envían los flujos de audio, secuencias de vídeo y otros datos. También podemos enviar comandos que controlan el flujo de los datos:
   - *Play*: Enviado por el cliente para iniciar un flujo.
   - *Play2*: A diferencia del comando *play* permite cambiar la tasa de bit de un flujo o reanudar un NetStream cuando se interrumpe la conexión.
   - *DeleteStream*: Se envía para borrar el NetStream.
   - *CloseStream*: Se envía para cerrar el NetStream.
   - *ReceiveAudio*: Para informar al servidor si enviar al cliente audio o no.
   - *ReceiveVideo*: Para informar al servidor si enviar al cliente vídeo o no.
   - *Publish*: Enviado por el cliente al servidor indicando un nombre al flujo. Cualquier cliente podrá recibir el audio, vídeo o los datos enviados.
   - *Seek*: Enviado por el cliente para indicar el tiempo en milisegundos de retardo en un fichero o lista de reproducción.
   - Pause: Pide al servidor parar o continuar reproduciendo.

[7]

2.2 IMS.

El estándar de IMS, “IP Multimedia Subsystem”, define una arquitectura genérica para ofrecer Voz sobre IP (VoIP) y servicios multimedia. Se trata de un estándar global, basado en conectividad IP y en una arquitectura del servicio de control que permite la utilización de diferentes tipos de servicios multimedia usando protocolos comunes basados en internet. En

primer lugar fue especificado por 3GPP/3GPP2 y actualmente está siendo aceptado por otros organismos de normalización como ETSI/TISPAN. [8]

Desde el punto de vista del usuario, los servicios basados en IMS permiten la comunicación entre usuarios o el acceso a contenidos (voz, texto, imagen y vídeo).

Arquitectura IMS.

Home Subscriber Server (HSS): Es la base de datos que contiene la información de los usuarios, es decir, contiene los suscriptores y sus servicios.

Call Session Control (CSCF). Son entidades encargadas de realizar tareas durante el registro y el establecimiento de sesión además del enrutamiento SIP. En una arquitectura IMS podemos encontrar tres CSCF, cada una con unas tareas específicas.

- Proxy CSCF (P-CSCF): Es el punto de acceso al núcleo IMS. Es el encargado de la compresión de los mensajes SIP, es responsable de la seguridad de los mensajes entre la red y el usuario y asegurar la veracidad de los mensajes SIP. Por otro lado, es el encargado de la negociación de los requerimientos de calidad, la gestión y control de las políticas de servicio. Determina que CSCF tendrá el control de la llamada.
- Interrogating CSCF (I-CSCF): Consulta al HSS y es el encargado de determinar el S-CSCF para un usuario, es decir, determina que S-CSCF será el encargado de la llamada.
- Serving CSCF (S-CSCF): Encargado del control de la sesión, autenticación del usuario y procesamiento de las llamadas. Puede haber más de un S-CSCF en una mima red. [9]

OpenIMSCore.

OpenIMSCore es un proyecto Open Source desarrollado por Fraunhofer Institute Fokus que ofrece los elementos básicos de una arquitectura IMS, funciones de control de sesión de llamada (CSCF's) y un servidor de subscripciones (HSS). [10]

Su instalación no es muy compleja. Se requiere de un equipo con Linux y una vez instalado su configuración es relativamente fácil. Además, ofrece una interfaz web para el HSS donde gestionar los usuarios.

2.3 Flash.

Flash es un complemento muy usado que se instala en los navegadores y permite mostrar contenidos visuales como audio o vídeo. Su uso es muy común y podemos verlo en anuncios de publicidad, juegos o simplemente para crear efectos visuales en algunas webs.

Para crear contenidos Flash se utiliza software como Adobe Flash Professional CC o Adobe Flash Builder. Compilado el código fuente su resultado es un fichero ejecutable de extensión SWF que será mostrado por el navegador.

Para la creación de aplicaciones web en Flash se utiliza ActionScript, un lenguaje de programación orientado a objetos muy similar a JavaScript diseñado para entornos como Adobe® Flash® Player y Adobe® AIR™. Se ejecuta en una máquina virtual, AVM (ActionScript Virtual Machine). Este lenguaje se compila en formato código generando un fichero SWF que se puede ejecutar con Flash Player o AIR. [11]

Flex

Apache Flex, anteriormente Adobe Flex, es un "framework" que permite desarrollar aplicaciones RIA (Rich Internet Applications). Flex compila la aplicación formada por documentos MXML para el diseño de interfaces gráficas y ActionScript para ejecutar la aplicación. El resultado es un archivo Flash de extensión SWF.

Es posible publicar el fichero SWF en una web que será mostrada por Flash Player o compilar con Adobe AIR para crear aplicaciones de escritorio. [12]

Red5 Server.

Red5 Server es un servidor Open Source que permite transmitir audio y vídeo en streaming. El proyecto es lanzado en Septiembre de 2005 y actualmente está disponible la versión 1.0. Esta desarrollado en Java y utiliza Flash para entregar el contenido. Usa RTMP o RTMPT como protocolo de comunicación en tiempo Real. [13]

Red5phone.

Red5Phone es una aplicación Open Source desarrollada en Flex que permite la transmisión de audio entre varios usuarios utilizando el servidor Red5. [14]

Deberemos descargar los ficheros necesarios para instalar esta aplicación en el lado del servidor Red5 y el código Flex con la aplicación en el servidor web. El código de la aplicación está desarrollado en Flex por lo que deberemos utilizar un software que sea capaz de compilar este código y genere un fichero SWF.

Figura 3: Interfaz aplicación Red5phone

El código Flex está disponible para todo el mundo y se puede descargar desde la web red5phone.googlecode.com. La aplicación está formada por diferentes ficheros. Los más

importantes son aquellos cuya extensión son MXML y AS. Los ficheros con extensión MXML (Macromedia eXtensible Markup Language) contienen principalmente el código de la interfaz gráfica aunque pueden contener código ActionScript. Por otro lado, los ficheros AS (ActionScript Source Code), contienen únicamente código ActionScript.

La estructura principal de un fichero MXML es:

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application
  xmlns:mx=http://www.adobe.com/2006/mxml
  layout="absolute"
  creationComplete="init()"
  preinitialize="presetup()"
  //Parámetros de la aplicación. (Dimensiones y estilo)
  width="266" height="262"
  alpha="1.0" borderStyle="solid" cornerRadius="10">

  <mx:Script>
          <![CDATA[

          //Código ActionScript

          ]]>
  </mx:Script>

  //Código con los diferentes elementos gráficos.

</mx:Application>
```

Toda aplicación en Flex debe estar definida dentro del elemento `mx:Application`. Se podrá incluir código ActionScript dentro del elemento `mx:Script`.

A continuación se listan los diferentes ficheros con una breve explicación de su contenido:

Ficheros MXML:

red5phone.mxml: Es el fichero principal de la aplicación. Contiene las variables necesarias para la comunicación con el servidor Red5. Además es el encargado de leer el fichero de configuración config.xml. Se definen dos elementos `local:LoginCanvas` y `local:PhoneCanvas`. Estos elementos de tipo local son objetos canvas definidos en los ficheros comentados a continuación.

LoginCanvas.mxml: Contiene el código Flex con los campos de texto para indicar el usuario y contraseña y el código ActionScript para la comunicación con el servidor Red5.

PhoneCanvas.mxml: Contiene los elementos gráficos de la aplicación como los botones del panel numérico, llamada, colgar, o barras del volumen de audio y micrófono. Además contiene el código con las funciones para el control de las llamadas (llamar, contestar, colgar…), la inicialización de los diferentes eventos (llamada entrante, saliente, perdida…), inicializar el micrófono, o establecer diferentes comandos RTMP (NetConnection, Publish,…).

Ficheros AS:

Red5Manager.as: Contiene la clase `Red5Manager`, con sus atributos, constructor y funciones. Los métodos que capturan las respuestas del servidor Red5 y las acciones SIP (Open, Call, HangUp, etc).

Red5MessageEvent.as: Contiene la clase encargada de los eventos producidos por el servidor Red5.

CallConnectedEvent.as y CallDisconnectedEvent.as: Contiene la clase encargada de los eventos producidos cuando se establece o se desconecta una llamada.

MissedCallEvent.as e IncomingCallEvent.as: Contiene la clase encargada de los eventos producidos cuando se recibe una llamada entrante o se produce una llamada perdida.

Otros:

assets/ : Directorio con los elementos multimedia utilizados, como sonidos o imágenes.

config.xml: Fichero XML utilizado para cambiar algún parámetro sin tener que compilar de nuevo la aplicación. Este fichero es leído por la aplicación al iniciar. Contiene el protocolo utilizado (RTMP) y la dirección IP del servidor Red5.

2.4 HTML5 y CSS3.

HTML5 es el nuevo estándar para HTML. Este estándar está aún en desarrollo. Sin embargo, la mayoría de los navegadores ya son compatibles y muchos de los nuevos elementos de HTML5 y sus APIs ya se pueden utilizar.

HTML5 nace de la colaboración entre el World Wide Web Consortium (W3C) y el Web Hypertext Application Technology Working Group (WHATWG). WHATTWG estaba trabajando en los formularios web y aplicaciones, mientras que W3C en XHTML 2.0. En 2006 deciden cooperar y crear una nueva versión de HTML. Las principales novedades son:

- Nuevas características que deben basarse en HTML, CSS, DOM y Javascript.
- Reducción de la necesidad de plugins externos (como Flash).
- Mejor manejo de errores.
- Más marcado para reemplazar scripting.
- HTML5 debe ser independiente del dispositivo.

Nuevas funciones.

Semántica: Se añaden nuevas etiquetas que no solo definen como mostrar el contenido sino qué es. Entre estas nuevas etiquetas podemos destacar:

- Nuevos elementos multimedia
- Nuevos elementos estructurales

- Nueva semántica de aplicación internacional
- Nuevos tipos de relaciones entre enlaces
- Nuevos atributos
- Nuevos tipos de formularios
- Nueva sintaxis de microdatos para ampliar la semántica

Funcionamiento sin conexión y almacenamiento: Nuevas APIs permiten a las aplicaciones web en HTML5 empezar con mayor rapidez o incluso funcionar sin conexión a internet (Application Cache). Como mejora a las cookies aparece Local Storage y Session Storage, permitiéndonos almacenamiento de datos en el navegador.

Aparece IndexedDB, para el almacenamiento de grandes cantidades de datos estructurados.

Acceso a dispositivos: HTML5 incluye nuevos APIs para ser utilizados en dispositivos móviles: vibración, geolocalización, acceso al audio y vídeo del micrófono y/o cámara, acceso a datos locales como contactos y eventos incluso a la orientación o inclinación.

Conectividad: Los Web Sockets y los eventos enviados desde el servidor permiten mayor eficiencia en la transmisión de datos entre el cliente y el servidor.

Multimedia: En HTML5, el audio y el vídeo se han convertido en las partes más importantes. Las nuevas API permiten manipular los estados de red y los datos cronológicos de los archivos, controlarlos y acceder a ellos.

3D, gráficos y efectos: Con SVG, Canvas, WebGL y las transformaciones 3D de CSS y el estándar SMIL, se pueden crear efectos visuales de forma nativa en el propio navegador en vez de utilizar como hasta ahora únicamente CSS y JavaScript para crear animaciones o efectos visuales, o tener que instalar complementos como Flash o SilverLight.

Rendimiento e integración: HTML5 proporciona nuevos APIs que permiten interactuar con datos binarios y con el sistema de archivos local del usuario, leer archivos de manera síncrona o asíncrona, arrastrar archivos del escritorio al navegador y escribir archivos en un directorio temporal.

Con los APIs de Web Workers y XMLHttpRequest 2 es posible que las aplicaciones web y el contenido dinámico se puedan ejecutar en segundo plano de manera independiente al resto del contenido, lo que nos proporciona mayor rapidez de carga y evitar esperas innecesarias.

CSS3: La nueva versión de CSS (hojas de estilo en cascada), CSS3 incluye nuevas tecnologías como las transformaciones en 2D, las transiciones y las transformaciones en 3D. Además Web Open Font Format (WOFF) proporciona mayor flexibilidad y control tipográfico [15], [16], [17].

HTML5 en dispositivos móviles.

Como hemos visto, HTML5 nos ofrece múltiples novedades. En la actualidad cada vez son más las personas que usan su móvil o Tablet para navegar por Internet. Los contenidos se muestren adecuadamente para estas dimensiones de pantalla es algo fundamental.

Por ello cada vez más se opta por un "diseño adaptativo", es decir, una web que se adapte a las dimensiones de la pantalla en la que se visualiza directamente, de manera que el contenido sea legible cómodamente y de manera ordenada.

2.5 WebRTC

WebRTC es un proyecto que permite la comunicación y transmisión de audio, vídeo y datos utilizando el navegador web sin la necesidad de plugins adicionales. Haciendo uso de diferentes APIs de Javascript se pueden realizar llamadas, compartir vídeo, el escritorio y enviar datos.

Está siendo llevado a cabo por dos entidades, Internet Engineering Task Force (IETF) y World Wide Web Consortium (W3C).

Aunque se está trabajando en desarrollar un estándar, la realidad es muy distinta y actualmente no todos los navegadores soportan esta nueva tecnología, o no soportan algunas de sus funciones como podemos ver en la siguiente tabla.

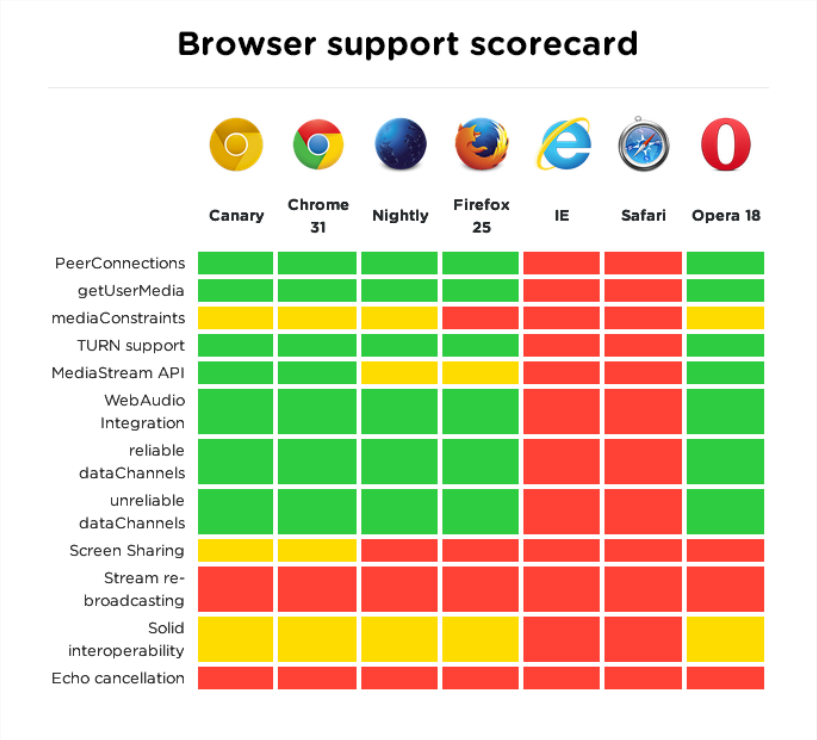


Figura 4: Soporte de webRTC según navegador.
Fuente: iswebrtcreadyyet.com

Arquitectura.

Principalmente hay dos escenarios posibles. Un primer escenario, el más sencillo, sería la comunicación entre dos navegadores, en el que dos usuarios se comunican entre ellos directamente desde una página web.

En la siguiente figura podemos ver dos equipos. Ambos hacen una petición HTTPS al servidor web donde se encuentra nuestra aplicación para abrir la aplicación en su navegador.

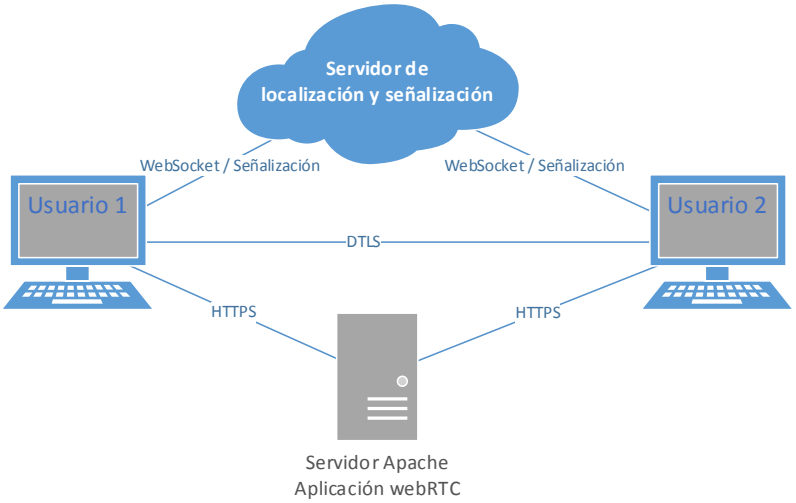


Figura 5: Topología de comunicación WebRTC entre dos navegadores

El segundo escenario sería cuando un usuario desde su navegador quiere comunicarse con otro usuario no conectado a la aplicación web. Para hacer esto posible es necesario un nuevo elemento en la arquitectura, un Gateway WebRTC el cual traduce a SIP.

En la figura mostrada a continuación podemos ver un usuario que realiza una petición HTTPS al servidor web para descargar la aplicación WebRTC en su navegador, esta vez se abrirá una comunicación websocket y la señalización a través del Gateway WebRTC que nos permitirá ponernos en contacto con cualquier cliente SIP conectado.

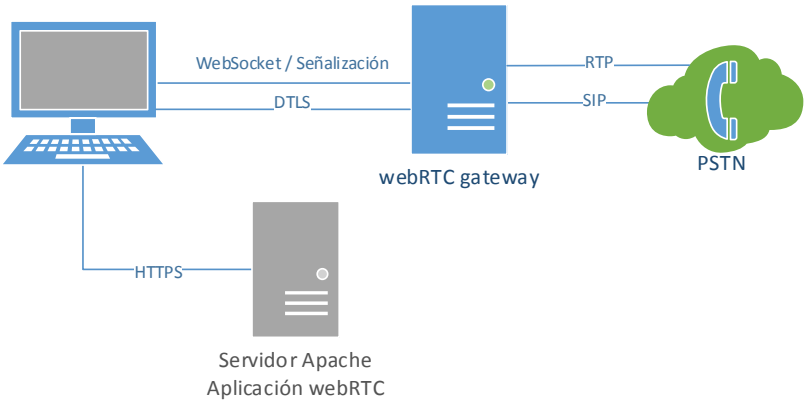


Figura 6: Topología de comunicación entre navegador red IMS/PSTN

Aspectos técnicos.

WebRTC permite transporte a través de NATS con ICE/UDP, ofrece seguridad utilizando el protocolo DTLS y control de congestión con SCTP. Uno de los inconvenientes es que no se define la forma de gestionar la señalización, pudiéndose hacer de diferentes maneras: SIP, XMPP, REST API, etc.

Los navegadores casi siempre se encuentran en equipos detrás de un NAT o de un cortafuegos. Esto hace muy difícil la comunicación entre los dispositivos y por ello se utiliza ICE, Interactive Connectivity Establishment. Utilizando los protocolos STUN y TURN, ICE permite descubrir y elegir qué dirección IP será utilizada para el intercambio de paquetes. [18]

La tecnología Session Traversal Utilities for NAT (STUN) permite a los equipos finales determinar qué dirección IP y puerto público le corresponde según la dirección IP y puerto privado asignados por NAT. [19]

Traversal Using Relay NAT (TURN), se utiliza cuando un equipo se encuentra tras un NAT y no es capaz de comunicarse con otros equipos. TURN además permite la comunicación con múltiples destinos usando la misma dirección de transmisión. [20]

ICE está basado en candidatos. Cada usuario descubre sus candidatos los cuales se intercambian en el paquete SDP. Hay diferentes tipos de candidatos, cada uno con mayor o menor prioridad.

- Candidatos HOST, son los candidatos con mayor prioridad y son aquellos con la ruta más corta, es decir, son los candidatos cuando dos equipos se encuentran en la misma red local.
- Candidatos REFLEX, tienen menor prioridad que los anteriores, son los candidatos cuando se requiere de una dirección IP externa fuera de la NAT facilitada por STUN.
- Candidatos RELAY, son los candidatos de menor prioridad, en este caso son utilizados cuando se necesita una dirección IP y puerto facilitado por el servidor TURN.

En cuanto a los codecs, WebRTC define una serie de codecs obligatorios que todos deben ser capaces de entender garantizando así la comunicación, aunque permite el uso de otros. G711A/G711u y Opus son los codecs de audio obligatorios mientras que VP8 es el códec de vídeo.

APIs y pasos para la comunicación entre dos equipos.

En la siguiente figura podemos distinguir entre el API destinado a los desarrolladores web que quieran implementar su aplicación utilizando WebRTC y los módulos correspondientes al API que será desarrollado por los navegadores.

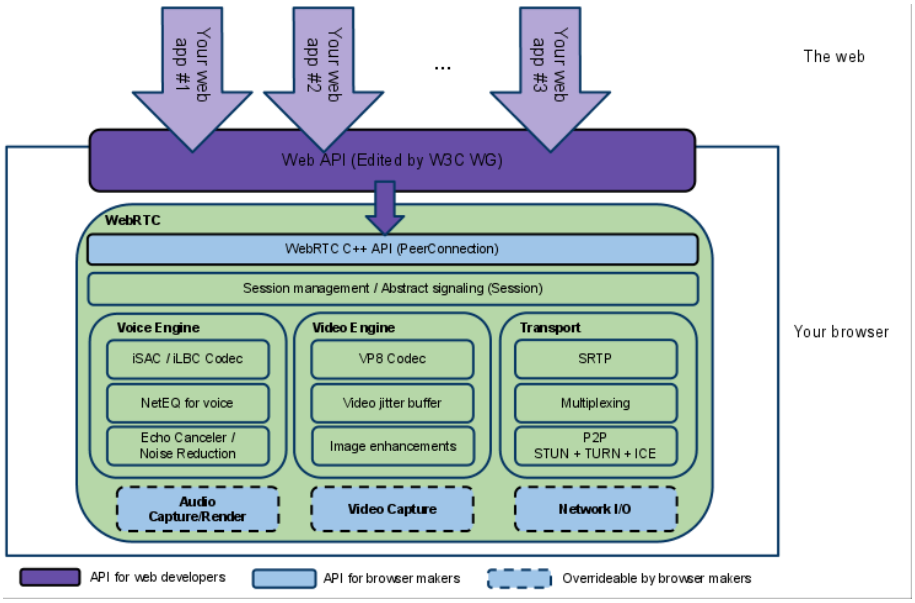


Figura 7: API WebRTC - Fuente: webrtc.org

WebRTC implementa tres APIs: MediaStream, RTCPeerConnection y RTCDataChannel. [21]

- MediaStream permite obtener acceso a los flujos de datos, como el audio del micrófono o el vídeo de la webcam. MediaStream está formado por varias pistas de audio o vídeo, representadas por el objeto MediaStreamTrack. Éstas pistas poseen una entrada, creada con getUserMedia() que obtiene el micro y la cámara del usuario y una salida que puede ser un elemento como <audio> o <video>, la PeerConnection API de WebRTC o una API de Audio Web MediaStreamAudioDestinationNode. [22]

- RTCPeerConnection: Realiza una conexión directa entre los navegadores para que puedan intercambiar datos entre ellos. Debido a que se encuentra en proceso de estandarización actualmente está implementado por Chrome como webkitRTCPeerConnection y por Firefox como mozRTCPeerConnection.

- RTCDataChannel: Permite enviar datos a través de la conexión PeerConnection. Un DataChanel es una conexión peer to peer (usuario a usuario) para transmitir cualquier tipo de datos. Incorpora mecanismos de seguridad, baja y permite alta capacidad. Sus usos son muy variados, desde videojuegos hasta chats en tiempo real, compartir ficheros o el escritorio, etc.

Los pasos a seguir para la comunicación entre dos dispositivos utilizando WebRTC son los siguientes. [23]

1. Obtener los flujos de datos. Con MediaStream se obtiene el audio del micrófono y/o el vídeo de la cámara.

2. Señalización. Se crea un PeerConnection y se envía una petición al segundo usuario (SDP). El segundo usuario recibe el SDP y generará y contestará al primero enviando un SDP de vuelta.

3. Buscar candidatos. El primer usuario recibe el SDP y genera sus candidatos ICE. Estos candidatos son enviados al segundo usuario el cual los procesa, genera los suyos propios y se los envía al primero.

4. Negociación de la sesión multimedia. Los usuarios se intercambiar contextos SDP con los codecs admitidos y acuerdan cuales utilizar.

5. Envío de datos. Iniciado el flujo RTCPeerConnection se envían por RTCDataChannel los flujos de audio y/o vídeo.

WebRTC2Sip.

WebRTC2SIP no es más que un Gateway que permite la comunicación entre WebRTC y el protocolo SIP, ofreciendo la posibilidad de realizar y recibir llamadas desde el navegador a una red SIP o PSTN. [24]

Arquitectura.

WebRTC2SIP está formado por cuatro módulos:

1. SIP Proxy: Encargado de la conversión del transporte SIP. Hace de interfaz entre la los equipos WebRTC y SIP.

2. RTCWeb Breaker: Permite la conversión y negociación de los flujos media entre WebRTC y redes SIP. El problema reside en que WebRTC soporte ICE y DTLS/SRTP mientras que las redes SIP no.

3. Media Coder: El estándar WebRTC define como obligatorios dos codecs de audio que todos deben implementar, opus y G711. El problema es la discusión actual de que codecs de vídeo utilizar, por lo que los navegadores pueden no coincidir. Media Coder permite la comunicación entre diferentes navegadores implementándolos todos.

4. Click-to-Call: Más que un módulo se trata de un servicio que permite con un simple click realizar una llamada.

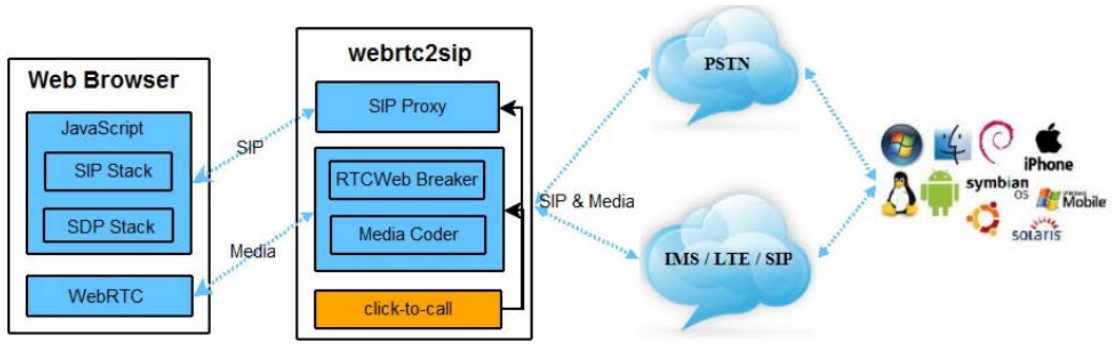


Figura 8: Arquitectura WebRTC2SIP - Fuente: webrtc2sip.org

Como podemos ver en la figura superior la arquitectura de WebRTC está formada por los cuatro módulos comentados anteriormente que comunican el navegador con la red SIP. En el navegador además del API WebRTC necesitaremos un cliente JavaScript. En este trabajo se ha decidido utilizar Sipml5.

Sipml5.

Sipml5 consiste en un cliente HTML5 SIP escrito completamente en JavaScript. Está formado por una pila SIP y una pila SDP que utilizan webSocket para el transporte y depende de WebRTC para la pila de media. [25]

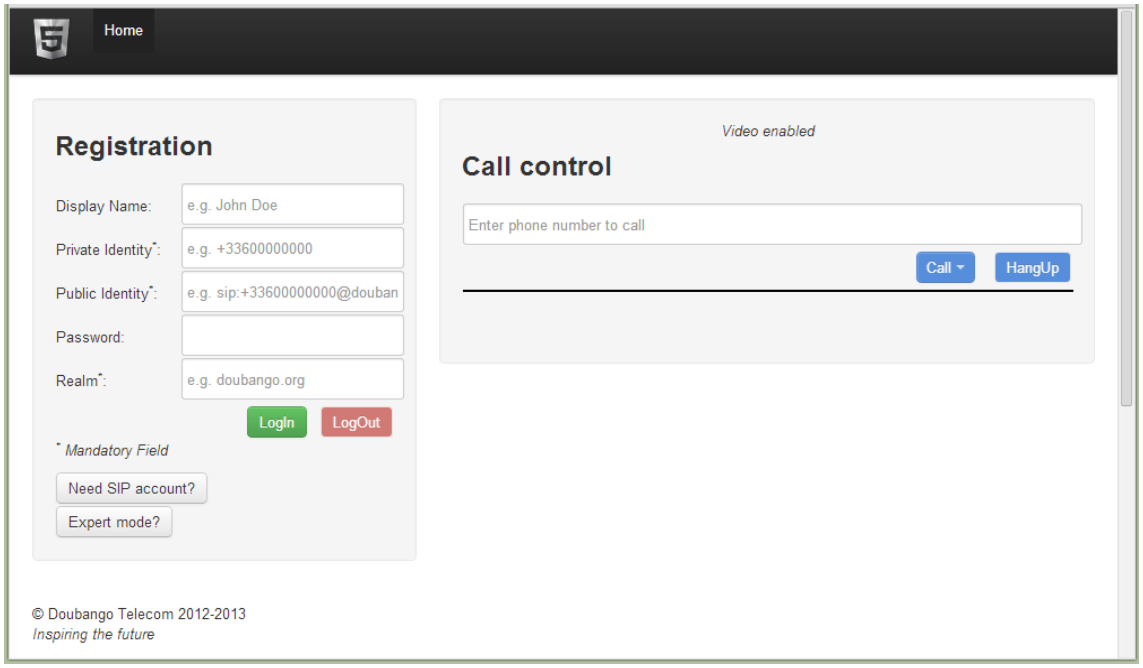


Figura 9: SipML5 live demo.

La figura superior muestra la web oficial con una demo del funcionamiento de sipML5.

Capítulo 3: Requisitos.

En este capítulo de listan los requisitos que debe cumplir la aplicación y una breve descripción de cada uno de ellos.

Aplicación Flash.

1. Autenticar al usuario.

   La aplicación deberá ser capaz de autenticar correctamente al usuario, notificándole en caso de error y no permitiendo el acceso sin un usuario y contraseña válidos.

2. Realizar y recibir llamadas a fijos, móviles y otros usuarios de la aplicación.

   El usuario podrá marcar el número desde el teclado o utilizando el panel de botones de la aplicación. El destino de la llamada podrá ser un número de teléfono tanto fijo como móvil o el número de otro usuario de la aplicación.

3. Enviar SMSs a números móviles y visualizar y borrar los SMSs enviados.

4. Visualizar las últimas llamadas entrantes y salientes desde la agenda.

   Se dispondrá de un registro de las llamadas donde el usuario podrá definir un nuevo alias a un contacto dado y realizar llamadas o enviar SMSs directamente a ese contacto pulsando únicamente un botón desde la agenda. Además se podrá borrar el registro si se desea.

5. Vídeo-llamada con otro usuario de la aplicación.

   Se podrá realizar una llamada de vídeo con otro usuario de la aplicación. El vídeo podrá ser mostrado por uno o ambos participantes de la llamada.

6. Valorar la llamada.

   Se podrá valorar mediante un formulario la calidad del audio, del vídeo y añadir algún comentario.

7. Cerrar sesión.

   La aplicación debe ser capaz cerrar la sesión del usuario cuando éste pulse sobre el botón dedicado a esta acción.

Aplicación HTML5.

La aplicación desarrollada en HTML5 sigue los mismos requisitos que la aplicación Flash con ligeras modificaciones, aunque como se comentó anteriormente el objetivo de este trabajo no se ha centrado en el desarrollo de la aplicación en HTML sino en la prueba de una alternativa que pueda ofrecer las mismas funcionalidades.

Los requisitos fundamentales que deberá realizar la versión HTML5 son:

1. Autenticar al usuario.
2. Realizar y recibir llamadas a fijos, móviles y otros usuarios de la aplicación.
3. Enviar SMSs a números móviles.
4. Vídeo-llamada con otro usuario de la aplicación.
5. Cerrar sesión.

Capítulo 4: Arquitectura del sistema.

En este capítulo se explica la arquitectura desplegada para el funcionamiento de la aplicación. Se comentarán aquellos equipos involucrados que se consideran más importantes y se indicarán los protocolos utilizados entre ellos. La intención de esta memoria no es entrar en detalle en algunos de estos equipos y su funcionamiento, por lo que se comentará de manera resumida.

Aplicación Flash.

En el despliegue de la aplicación Flash nos encontramos en primer lugar con el equipo del usuario. Este equipo será un PC con conexión a internet que realizará una petición a la web donde se encuentra la aplicación.

Esta petición es enviada al equipo (llamado Uphone en la figura) donde se encuentran instalados varios servicios. Como vemos en la figura 10, consta de dos elementos, aunque se ha utilizado un mismo equipo éstos pueden ser instalados en equipos diferentes.

- Servidor Apache, es el servidor web que permite al usuario acceder a la web y descargar la aplicación en su navegador. Las peticiones realizadas a este servidor pueden ser HTTP o HTTPS como en este caso, ofreciendo mayor seguridad al ir éstas peticiones cifradas.
- Servidor Red5, es el servidor Flash que permite la comunicación con la aplicación utilizando el protocolo RTMP. Muchos cortafuegos o firewalls bloquean este tipo de tráfico por lo que se ha usado RTMPT (RTMP Tunelado).

Cuando el usuario desea enviar un SMS, se realiza una petición al servidor Apache. Esta petición, es recogida y procesada por una página web escrita en PHP (ver capítulo 7). Procesada la petición enviada por el cliente es reenviada a otro equipo en el que se encuentra un servicio llamado Kannel, un gateway SMS que permite el envío de SMS. Utilizando el protocolo SMPP (Short Message Peer-to-Peer) se comunica con un SMSC (Short Message Service Center) encargado de enviar los SMSs a la red móvil.

El servidor Flash, Red5Server, “traduce” los mensajes RTMPT enviados por el cliente a SIP y éstos son enviados a otro equipo, el SBC (Session Border Controler). Este equipo se ha utilizado como punto de seguridad, ya que permite ocultar la topología que se encuentra al otro lado, evita ataques de denegación de servicio (DoS) y ofrece calidad de servicio (QoS). Aunque su uso es prescindible, ya que el sistema funcionaría sin él, se ha utilizado para simular lo más posible el despliegue en un entorno de producción.

A continuación se encuentra el CoreIMS. Este equipo contiene instalado el CoreIMS de Fokus, OpenIMSCore explicado en el capítulo 2.2. Los sistemas que conforman el sistema IMS están virtualizados en este equipo, excepto el PCSCF cuyo papel lo desempeña el SBC.

Por último se utiliza un Voice Gateway, en concreto el modelo Cisco SPA3102. Se trata de un dispositivo al que se le ha conectado una línea de teléfono y que permite conectar con la red PSTN. De esta manera las llamadas cursadas por los usuarios y que utilizan SIP pueden salir a la red de telefonía y comunicarse con números de teléfono fijos y móviles.

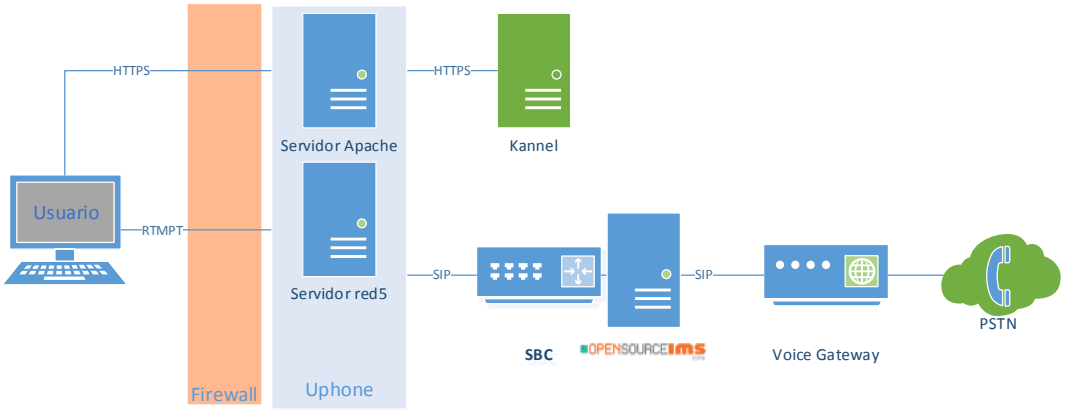


Figura 10: Topología aplicación Flash

Aplicación HTML5

El despliegue para la versión HTML5 es muy parecida a la utilizada para la aplicación Flash. Al igual que en el caso anterior se dispone del equipo llamado *Uphone* donde tenemos el servidor web Apache pero en vez de tener el servidor Flash, en este caso tendremos el gateway WebRTC2SIP. Éste nos permite establecer la comunicación webSocket y el flujo media RTP (Real-time Transport Protocol) generada por la aplicación con el SBC, traduciendo a SIP. Además se ha utilizado el servidor STUN ofrecido por google para conocer la dirección IP del equipo si este se encuentra dentro de un NAT. Consulte el capítulo 2.5 para más detalles sobre el funcionamiento de WebRTC y WebRTC2SIP.

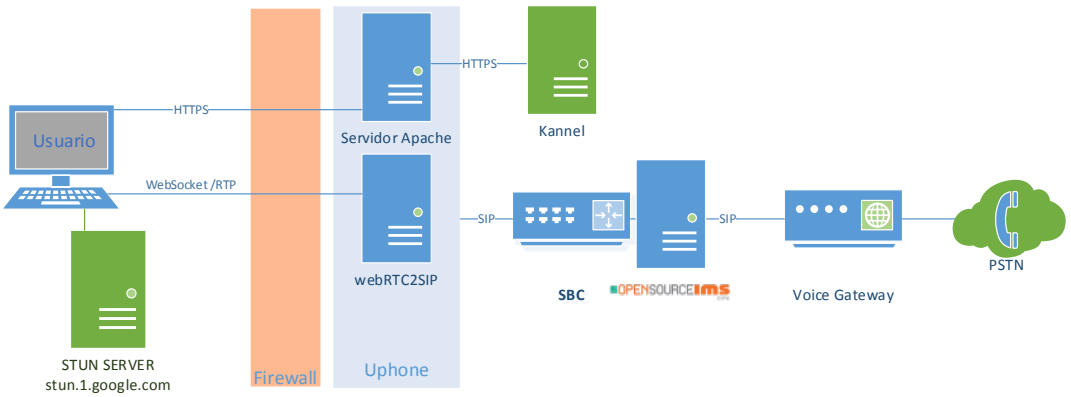


Figura 11: Topología aplicación HTML5

Capítulo 5: Servicio con cliente Flash.

En este capítulo se explicará el desarrollo de la aplicación en Flash, las decisiones tomadas y tecnologías utilizadas. La aplicación se ha desarrollado a partir del código fuente de Red5phone añadiendo y modificando en gran parte su estructura y funcionalidades.

Diseño gráfico.

El diseño de la aplicación se ha creado totalmente desde cero. Para ello se ha utilizado Adobe Photoshop para la creación de las imágenes y los diferentes elementos de la interfaz aunque se han utilizado los iconos y tipografía oficiales de Telefónica. Los principales colores son el azul y el verde todos ellos colores significativos de la compañía.

La aplicación simula la pantalla de un terminal móvil ya que las funcionalidades que nos ofrece son las que podríamos realizar con uno real. En la parte superior de la aplicación, la cabecera, se muestra un panel que indica al usuario el estado en el que se encuentra y le servirá al usuario para saber qué acción llevar a cabo o qué está sucediendo. Debajo de la cabecera está el cuerpo de la aplicación, donde el usuario podrá interactuar introduciendo datos o se le mostrará el contenido correspondiente. En la parte inferior dentro del cuerpo se dispondrá de los diferentes botones para manejar la aplicación. Por último, en la parte inferior se muestran cuatro botones que forman el menú. Este menú permite cambiar entre los estados principales.

Figura 12: Diseño aplicación Flash

Los diferentes estados permiten al usuario moverse entre las diferentes funcionalidades que ofrece la aplicación y mostrar la información de lo que está ocurriendo en un momento dado.

Pantallas o estados de la aplicación.

La implementación de los diferentes estados no es muy compleja. En el fichero mxml, en nuestro caso `PhoneCanvas.mxml`, se definen etiquetas `<mx:Canvas>`. Una etiqueta canvas es un lienzo donde se colocan los diferentes elementos gráficos que formarán el estado. Los diferentes estados se controlan con la etiqueta `<mx:states>`, definiendo en su interior tantas etiquetas `<mx:State>` como estados tengamos. Dentro de cada `<mx:State>` se pueden modificar las propiedades de los elementos gráficos, y por lo tanto se utilizará para decidir que canvas mostrar:

A continuación se muestra un ejemplo del esquema explicado con dos estados cada uno con un botón:

```
<mx:Canvas id="estado1" visible="true" x="0" y="0" width="250" height="377"
alpha="1.0">
<mx:Button id="boton1"   x="113" y="225" width="111" height="35"
click="funcion1();"/>
</mx:Canvas>
<mx:Canvas id="estado2" visible="false" x="0" y="0" width="250" height="377"
alpha="1.0">
<mx:Button id="boton2"   x="113" y="225" width="111" height="35" click="
funcion2();"/>
</mx:Canvas>

<mx:states>
<mx:State name="estado1">
//Mostramos el estado actual y ocultamos el resto
<mx:SetProperty name="visible" target="{estado1}" value="true"/>
<mx:SetProperty name="visible" target="{estado2}" value="false"/>
</mx:State>
<mx:State name="estado2">
//Mostramos el estado actual y ocultamos el resto
<mx:SetProperty name="visible" target="{estado1}" value="false"/>
<mx:SetProperty name="visible" target="{estado2}" value="true"/>
</mx:State>
//Resto de los estados, si hubiera más.
<mx:states>
```

Definidos los estados en el mxml se ha definido una función en ActionScript que nos permite cambiar de uno a otro:

```
  private function changeState(state:String):void{
            currentState = state;
//Código donde se decide que hacer en cada estado.
}
```

La aplicación consta de un total de 13 estados: inicio, pantalla principal, llamada entrante, llamada saliente, llamada establecida, video-llamada con vídeo saliente, video-llamada con vídeo entrante, video-llamada con vídeo en ambos sentidos, agenda, lista de SMSs enviados, enviar SMS, formulario y saliendo de la aplicación.

En la figura 13 se muestra un diagrama con todos los estados y la relación entre ellos.

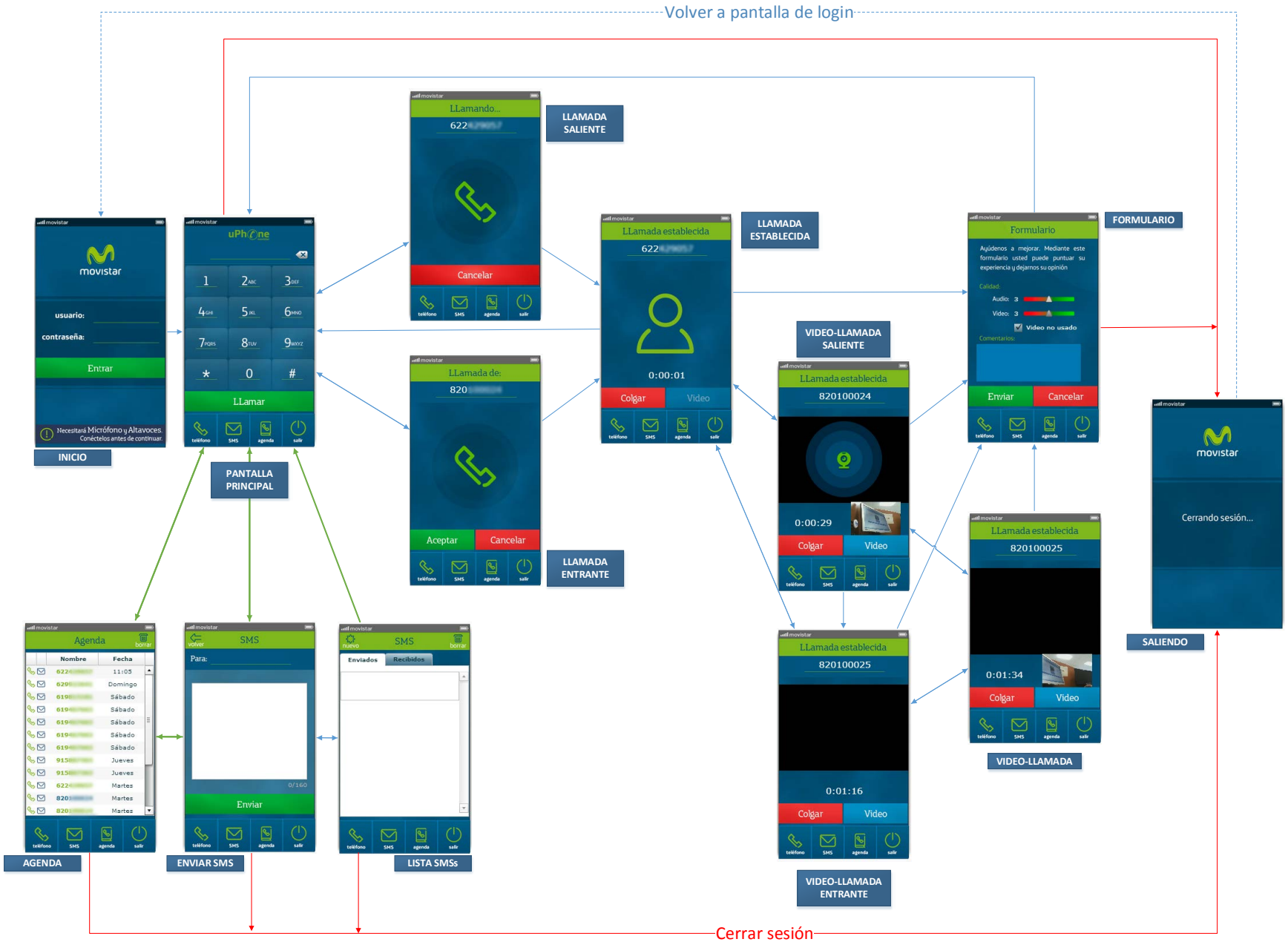


Figura 13: Diagrama de estados

Micrófono y cámara.

ActionScript nos permite acceder al micrófono y la cámara a través de su API utilizando las clases Microphone y Camera. Cada una de estas clases tiene sus propios métodos y atributos que nos permiten inicializar y obtener el sonido e imagen de los dispositivos conectados además de cambiar algunos de sus parámetros como la ganancia, el códec utilizado, etc.

Para obtener el micrófono utilizaremos `getMicrophone`:

```
public function initMicrophone():void {
  mic = Microphone.getMicrophone();

  if(mic == null)
  {
    //Si no hay micrófono conectado
  }else{
    //Inicializamos el micrófono con los parámetros deseados.
    mic.setUseEchoSuppression(true);
    mic.setLoopBack(false);
    mic.setSilenceLevel(0,20000);
    mic.gain = 60;
    mic.rate = 8;
    mic.addEventListener(ActivityEvent.ACTIVITY,    micActivityHandler);
    mic.addEventListener(StatusEvent.STATUS, micStatusHandler);
  }
}
```

Para obtener la cámara utilizaremos `getCamera`:

```
public function initCamera():void {
  cam = Camera.getCamera();
  if(cam == null){
    //Si no hay cámara conectada
  }else{
    //Inicializamos el micrófono con los parámetros deseados.
    cam.setKeyFrameInterval(5 );
    cam.setMode(320,240, 15 );
    cam.setQuality(0, 100 );
  }
}
```

Enviar y recibir audio y vídeo.

En el capítulo 2.2 se habla sobre RTMP y los comandos. En este apartado veremos cómo utilizar estos comandos desde ActionScript para comunicarnos con el servidor Flash y poder transmitir y recibir los flujos media. Para ello se hace uso de las clases `NetConnection`, encargada de crear la conexión entre el cliente y el servidor y `NetStream`, encargada de abrir el canal de transmisión sobre un NetConnection. Es importante tener encuentra que NetConnecion creará una conexión bidireccional mientras que NetStream es unidireccional teniendo que crear dos flujos si queremos recibir y enviar al mismo tiempo. [26] [27]

- Cómo enviar nuestro flujo media al servidor Flash:

  1. Crear un objeto NetConnection y llamar a NetConnection.connect() para conectarnos al servidor.

```
public  var netConnection:NetConnection = null;
private var outgoingNetStream:NetStream = null;

netConnection = new NetConnection();
netConnection.connect(red5Url);
```

  2. Añadir un escuchador al objeto NetConnection con NetConnection.addEventListener() para recibir eventos NetStatusEvent.

```
netConnection.addEventListener( NetStatusEvent.NET_STATUS , netStatus );
```

  3. Cuando se reciba un evento "NetConnection.Connect.Success" crear el NetStream pasando el objeto NetConnection como parámetro al constructor.

```
outgoingNetStream = new NetStream(netConnection);
```

  4. Capturar el audio y el vídeo con los métodos NetStream.attachAudio() y NetStream.attachCamera().

```
outgoingNetStream.attachAudio(mic);
outgoingNetStream.attachCamera(cam);
```

  5. Emitir el flujo con el método NetStream.publish().

```
outgoingNetStream.publish(publishName, "live");
```

- Cómo reproducir un flujo enviado por el servidor Flash:

  1. Crear un objeto Video y un VideoDisplay donde se mostrará. Llamar a la función addChild() para añadir el vídeo a la pantalla.

```
<mx:VideoDisplay id="pantallaVideo"/>

public var videoRemoto :Video = new Video();
pantallaVideo.addChild(videoRemoto);
```

  2. Crear un objeto NetConnection y llamar a NetConnection.connect() para establecer la conexión con el servidor.

```
public  var netConnection:NetConnection = null;
private var incomingNetStream:NetStream = null;

netConnection = new NetConnection();
netConnection.connect(red5Url);
```

  3. Añadir un escuchador al objeto NetConnection con NetConnection.addEventListener() para recibir eventos NetStatusEvent.

```
netConnection.addEventListener( NetStatusEvent.NET_STATUS , netStatus );
```

4. Cuando se reciba un evento "NetConnection.Connect.Success" crear el NetStream pasando el objeto NetConnection como parámetro al constructor.

```
incomingNetStream = new NetStream(netConnection);
```

5. Llamar al método attachNetStream() del objeto Video pasando el objeto NetStream.

```
videoRemoto.attachNetStream(incomingNetStream);
```

6. Llamar al método play() del objeto NetStream para empezar a mostrar el flujo.

```
incomingNetStream.play(publishName);
```

- Cómo cerrar el flujo:

Para cerrar el flujo media, ya sea el que estamos emitiendo o reproduciendo deberemos utilizar el objeto NetStream creado. En el caso del flujo recibido cancelaremos la reproducción mediante el método NetStream.play() con parámetro false. Por el contrario si estamos emitiendo, deberemos primero liberar el flujo de los medios capturados (micrófono y cámara), liberar la pantalla del objeto Video y por último cerrar el flujo mediante el método NetStream.close().

```
  if(incomingNetStream != null) {
          incomingNetStream.play(false);
  }
  if(outgoingNetStream != null) {

          outgoingNetStream.attachAudio(null);
          outgoingNetStream.attachCamera(null);

          videoRemoto.attachNetStream(null);

          outgoingNetStream.close();
  }
```

Comunicación con el servidor Red5.

Durante el uso de la aplicación puede interesarnos comunicarnos con el servidor Red5 para que este realice alguna acción. Por ejemplo, podemos enviar una notificación al servidor y que este se la transmita a un usuario determinado.

A continuación se explica cómo se implementa tomando como ejemplo la situación en la que un usuario envía el vídeo a otro y a éste se le muestra en su pantalla.

En primer lugar definimos la función que se ejecutará cuando el usuario pulse el botón de enviar el vídeo.

```
public function msgVideotoRed5():void {
   parentApplication.red5Manager.doVideoAlert(numeroDestino);
   doBotonVideo(true);
}
```

En el código anterior se define una función que realiza una llamada a la función de la clase Red5Manager, *doVideoAlert* indicándole el número del usuario con el que estamos comunicados. En segundo lugar se llama a la función *doBotonVideo* que es la encargada de realizar los comandos RTMP necesarios para establecer los flujos de vídeo. El parámetro de esta función sirve de *flag* para controlar quien inicia el vídeo.

```
        public function doVideoAlert(dest:String):void {
                netConnection.call("videoAlert", null, uid, dest);
        }
```

La función *doVideoAlert* es la encargada de realizar la petición al servidor red5. Para ello se utiliza la función netConnection.call(). El método call() envía un comando RTMP en el servidor Flash. Los parámetros de esta función son el comando que queremos enviar, la respuesta y argumentos adicionales (en nuestro caso el identificador de usuario y el número de destino).

El servidor Red5 recibirá el comando *videoAlert* y enviará otro comando al usuario destino notificándole para que se le muestre el vídeo. En este trabajo no se explica el código de la parte del servidor, únicamente el código del cliente Flash.

El usuario destino deberá poder recibir el comando que le envíe el servidor Red5 (videoAlerta), para ello creamos un evento que escuche a esta notificación:

```
public function videoAlerta():void
{
  dispatchEvent (new Red5MessageEvent(Red5MessageEvent.MESSAGE,
"videoAlerta", "Videollamada entrante"));
}
```

Por último deberemos decidir que realizar cuando nos llega este mensaje utilizando la función desarrollada por Red5phone:

```
private function receivedRed5MessageEvent(event:Red5MessageEvent):void {
  var msgType:String = event.msgType;
  var message:String = event.message;
  switch(msgType) {
          case Red5MessageEvent.NETSTAUS:
  //…código…
          break;
          case Red5MessageEvent.SIP_REGISTER:
  //…código…
          break;
  //…código…
          case "videoAlerta":
                  phoneCanvas.doBotonVideo(false);
          break;
          default:
  //…código…
  }
}
```

Peticiones al servidor WEB.

La aplicación realiza peticiones al servidor web para llevar a cabo algunas funcionalidades como registrar las llamadas o enviar un SMS. Cómo se recogen estas peticiones en el servidor se explicará en el capítulo 7.

Para realizar peticiones al servidor web desde ActionScript deberemos utilizar la etiqueta HTTPService en el código MXML.

```
<mx:HTTPService
  id="nombreService"
  result="funcionPeticion(event)"   //Función a realizar
  resultFormat="e4x"                //"e4x" para obtener un fichero XML.
                                    //"text" para una petición web.
  showBusyCursor="true"             //Cambia el icono del cursor
  url="dirección web"               //Dirección de la página o recurso
  method="POST">                    //La petición puede ser POST o GET
</mx:HTTPService>
```

Como vemos en el código anterior definimos un objeto HTTPService con un nombre y se le asigna una función. En esta función definiremos que queremos enviar en la petición POST al servidor.

```
private function funcionPeticion():void
{
  var parametros:Object = {};
  parametros ["item1"]    = "texto1";
  parametros ["item2"]    = "texto2";

  //Añadimos eventos para saber si la petición se realiza con éxito.
  nombreService.addEventListener("result", respuestaExito);
  nombreService.addEventListener("fault", respuestaFallo);

  //Podemos modificar los atributos del objeto HTTPService en la función
  httpservSetNewAlias.method = "POST";
  httpservSetNewAlias.useProxy = false;

  //Se envía la petición.
  nombreService.send(parametros);
}
```

Los eventos asignados a la petición para controlar la respuesta, es decir saber si la petición se ha realizado con éxito o no:

```
public function respuestaFallo (event:ResultEvent):void
{  //Código cuando la petición falle.
}

public function respuestaExito (event:ResultEvent):void
{
  //Código cuando la petición se ha realizado con éxito.
  // Podemos controlar la respuesta recibida
  if (event.result.Response == "textoRespuesta")
  //Código si el servidor nos responde con la cadena "textoRespuesta".
}
```

Ejecutar código JavaScript desde ActionScript.

ActionScript nos permite hacer llamadas a funciones JavaScript de la página web donde se encuentra la aplicación Flex. Para ello se utiliza la clase ExternalInterface y se utiliza el método call():

```
//Llamada a funcionJavaScript con parámetro “param”
ExternalInterface.call("funcionJavaScript", param);

//Llamada a funcionJavaScript sin parámetros y guardamos lo que
devuelva.
var texto:String = ExternalInterface.call("funcionJavaScript ")
```

Uno de los usos de ejecutar código JavaScript desde la aplicación ha sido el poder avisar al usuario de una llamada entrante mediante ventanas emergentes cuando el usuario tiene minimizado el navegador, o recargar la web cuando se cierra sesión.

Capítulo 6: Servicio con cliente HTML5 y WebRTC.

En este capítulo se explicará la implementación de la aplicación web utilizando HTML5, JavaScript, diferentes librerías y APIs además de WebRTC. Se ha desarrollado utilizando el código fuente de Sipml5.

El desarrollo de la aplicación HTML5 ha consistido principalmente en la instalación de Webrtc2sip y Sipml5. Modificando posteriormente el diseño de la web de prueba de Sipml5 para que ofreciese el mismo aspecto y mismas funcionalidades que la versión Flash desarrollada anteriormente.

Figura 14: Aplicación HTML5 en escritorio

Pantallas o estados de la aplicación

Al igual que la aplicación Flash esta versión está formada por diferentes estados que conforman la aplicación.

La aplicación está organizada en diferentes bloques de código, cada estado está escrito en HTML dentro de un div. Un div es un elemento utilizado en HTML para definir una división o sección en una web, también es utilizado para organizar o agrupar varios elementos y poder manejarlos conjuntamente con CSS.

Para cambiar de un estado a otro se ha definido una función en JavaScript similar a la función desarrollada en la aplicación Flash. Esta vez utilizamos la propiedad CSS `display` para mostrar u ocultar los diferentes estados.

```
function cambiaEstado (estado, subEstado)
{
        if(estado == "estado1")
        {
           document.getElementById("divEstado1").style.display='block';
           document.getElementById("divEstado2").style.display='none';
        }

        if(estado == "estado2")
        {
           document.getElementById("divEstado1").style.display='none';
           document.getElementById("divEstado2").style.display='block';

           switch(subEstado)
           {
            case "subEstado1":
           {
            //Código que se ejecuta en el estado2, subestado1

           }break;

            case "subEstado2":
           {
            //Código que se ejecuta en el estado2, subestado2

           }break;

            default:
           {
            //Código que se ejecuta en el estado2 por defecto.

           }
           }
        }
}
```

El código anterior corresponde a una función que permite ocultar o mostrar diferentes divs (diferentes estados de la aplicación) pasando por parámetros el nombre del estado y un subestado si queremos que para un mismo estado se realicen diferentes acciones.

Peticiones al servidor WEB:

Como se comentó anteriormente en el desarrollo de la versión en Flash, la aplicación realiza diferentes peticiones al servidor web para realizar algunas acciones como enviar SMSs o registrar las llamadas.

En el cliente Flash realizábamos estas peticiones mediante la clase HTTPService de ActionScript. Ahora, desde la versión HTML5, estas peticiones se realizan desde JavaScript utilizando XMLHttpRequest.

Ejemplo:

```
function enviaPeticion(){

  if (window.XMLHttpRequest){// codigo para IE7+, Firefox, Chrome…
          xmlhttp=new XMLHttpRequest();
  }else{// codigo para IE6, IE5
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
          xmlhttp.open("POST", "./fichero.php",true);
          xmlhttp.setRequestHeader("Content-type","application/x-www-
form-urlencoded");
          xmlhttp.send("parametro="+value);

}
```

El código anterior es un ejemplo de cómo enviar una petición POST a un fichero PHP alojado en el servidor web. En primer lugar se crea un objeto XMLHttpRequest. Posteriormente con el método open se define el tipo de petición (GET o POST) y el fichero PHP, después se indican las cabeceras de la petición (en este caso se indica que los datos a enviar serán enviados como un formulario) y por ultimo con la función send() se indican los datos y envía la petición.

Almacenamiento de datos en el cliente.

Como se comentó en el capítulo 2.4, HTML5 ofrece nuevas maneras de guardar. Application Cache, para almacenar datos y poder acceder a ellos sin conexión, Local Storage y Session Storage para guardar datos en el propio navegador e IndexedDB para el almacenamiento de datos estructurados.

Para el desarrollo de la aplicación en HTML5 se ha utilizado LocalStorage para almacenar datos, como el nombre de usuario o algunos parámetros necesarios por la pila sipml5. Su uso es muy sencillo. Mediante código JavaScript utilizaremos los métodos getItem() y setItem() para leer o guardar datos. A continuación vemos un ejemplo:

```
if (window.localStorage) {
  window.localStorage.setItem('nombre', txtDisplayName.value);
}
```

En este código utilizamos el almacenamiento local (Local Storage) para guardar el valor de un campo de texto. Este tipo de almacenamiento es tipo clave-valor, es decir, se debe asignar un nombre al contenido guardado. En este caso guardamos como clave “nombre” el valor del campo.

```
if (window.localStorage) {
  var miNombre;
  miNombre = window.localStorage.getItem('nombre');
}
```

Para leer se utiliza getItem con la clave utilizada para guardar ese valor. El código anterior guarda en la variable *miNombre* el valor con la clave *nombre*.

Notificaciones

Las notificaciones web es uno de los nuevos APIs de HML5. Permite mostrar al usuario una notificación de escritorio mediante código JavaScript, pero para que una web pueda mostrar notificaciones, deberá tener los permisos del usuario.

El siguiente código muestra cómo utilizar las notificaciones en Chrome:

```
//Pedir permisos al usuario para mostrar notificaciones.
  webkitNotifications.requestPermission();

  var notification = webkitNotifications.createNotification('imagen',
'Título', contenido);

  notification.show();   //Muestra la notificación
  notification.cancel(); //Cierra la notificación

  notification.onclick = function(){
          //Qué hacer cuando el usuario pulsa sobre la notificación.
  };
```

Cliente para dispositivos móviles en HTML5:

Se ha desarrollado una versión similar a la anterior, pero adaptada para dispositivos táctiles y con pantallas de menores dimensiones.

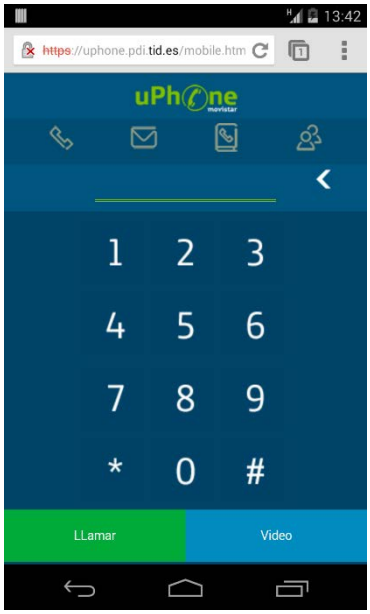


Figura 15: Aplicación HTML5 en móvil

Diseño de la aplicación.

El diseño para dispositivos móviles es muy similar a la versión de escritorio. Siguen predominando los colores azules y verdes característicos de Telefónica, así como el uso de la iconografía oficial de la compañía.

La organización de la aplicación ha sido ligeramente modificada para mejorar la experiencia de usuario.

Al igual que en la versión de escritorio en la parte superior de la aplicación, la cabecera, se muestra un panel que indica al usuario el estado en el que se encuentra y que le servirá para saber qué acción llevar a cabo o qué está sucediendo. Una modificación respecto a la versión anterior es que debajo de la cabecera ahora se muestra el menú con los botones que permitirán cambiar entre los estados principales. Debajo de éstos se encontrará el cuerpo de la aplicación, donde el usuario podrá interactuar introduciendo datos o se le mostrará el contenido correspondiente. En la parte inferior dentro del cuerpo se dispondrá de los diferentes botones para manejar la aplicación.

Eventos touch en JavaScript.

Una de las diferencias principales entre la versión de escritorio y la móvil es que esta última será manejada con terminales táctiles. Por ello los eventos utilizados en los botones no tienen por qué ser iguales.

El evento más común de un botón en JavaScript es el evento *onclick* con el cual podemos controlar que ocurre cuando el usuario presiona dicho botón. [28] Este evento pensado para ser utilizado con ratón no funciona bien del todo cuando se utiliza en pantallas táctiles y por ello se definen un nuevo tipo de eventos, los eventos Touch (touchstart, touchend, touchmove y touchcancel). [29]

A continuación se muestra un ejemplo de cómo aplicar un evento touch a un botón para que al pulsarlo añada un 1 al texto de un campo.

```
var c1 = document.getElementById('char1');
c1.addEventListener('touchstart',function(event) {
                campoTexto.value = campoTexto.value+"1";}, false);
```

El código anterior escrito en JavaScript define una nueva variable que obtiene el elemento HTML que corresponde al botón con id 'char1' para posteriormente asignarle un evento con la función addEventListener pasado por primer parámetro y la función a ejecutar cuando se de dicho evento.

Vibración.

Otra de las características que ofrece un terminal móvil es la de notificar mediante vibraciones. Gracias al nuevo API de vibración ahora es posible hacer vibrar nuestro dispositivo desde el código JavaScript. [30]

A continuación se muestra un ejemplo de cómo añadir un evento a un botón para que vibre 20ms cuando se pulse:

```
var timVibrate = 20; //tiempo de vibración [ms]

var c1 = document.getElementById('char1');
c1.addEventListener('touchstart',function(event){
         if(navigator.vibrate) navigator.vibrate(timVibrate);}, false);
```

Capítulo 7: Recepción, procesado y gestión de datos.

En este capítulo se explicarán las funcionalidades comunes en ambas aplicaciones del lado del servidor web, es decir, cómo se procesan las peticiones realizadas al servidor Apache y la estructura de los ficheros generados. Además se explicará el diseño de las diferentes webs donde visualizar los registros y la gestión de los usuarios.

Servidor Apache y PHP.

La aplicación realizará diferentes peticiones al servidor web Apache instalado. Estas peticiones explicadas en los capítulos 5 y 6 se realizan a diferentes páginas web escritas en PHP que procesan esta petición y devuelven un resultado.

La aplicación web desarrollada utiliza ficheros PHP y diferentes ficheros XML y un fichero JSON para guardar y generar registros de las llamadas, SMSs, cuestionarios o recopilar estadísticos.

7.1 Archivos XML

A continuación se detallan los diferentes archivos XML, su uso y estructura.

Agenda contiene los ficheros XML con los registros de las llamadas individualmente por usuario. Este directorio contiene dos tipos de archivo base.xml y [nombre usuario].xml

Base.xml contiene la raíz del fichero XML y se utiliza como plantilla cuando no existe un fichero XML para el usuario y [Nombre usuario].xml contiene el registro de las llamadas del usuario en cuestión. Consta de los siguientes campos:

Tabla 1 Campos de una entrada en la Agenda

CAMPO	DESCRIPCIÓN
ID	Identificador de usuario. Número de teléfono o usuario SIP.
ALIAS	Alias que le asigna un usuario a un nombre. Por defecto será el mismo que el id hasta que el usuario lo cambie.
HORA	[HH:MM]
DÍA	[lunes, martes, miércoles, jueves, viernes, sábado, domingo]
FECHA	[DD-MM-YYYY]
ESTADO	[true, false] Indica si la llamada es entrante o saliente.

Dependiendo de la antigüedad del registro de llamada en la aplicación se mostrará un campo u otro. Si el registro es de hoy se mostrará la hora, si es de esta semana se muestra el día y si es más antiguo la fecha.

Ejemplo de un fichero XML de agenda:

```
<Agenda>
  <entrada id="622622622">
          <alias>Bob</alias>
          <hora>11:23</hora>
          <dia>Jueves</dia>
          <fecha>09-05-2013</fecha>
          <estado>true</estado>
  </entrada>
  <entrada id="633633633">
          <alias>Alice</alias>
          <hora>11:23</hora>
          <dia>Viernes</dia>
          <fecha>10-05-2013</fecha>
          <estado>false</estado>
  </entrada>
</Agenda>
```

SMS contiene el registro de los SMSs enviados individualmente por un usuario en concreto. Contiene dos tipos de archivo: base.xml y [nombre usuario]-send.xml:

Base.xml contiene la raíz del fichero XML. Se utiliza como plantilla cuando no existe un fichero XML para el usuario y [Nombre usuario]-send.xml contiene un registro con los SMSs enviados por el usuario. Sus campos son los siguientes:

Tabla 2 Campos de una entrada en SMS

CAMPO	DESCRIPCIÓN
ID	Identificador de usuario. Número de teléfono o usuario SIP.
ALIAS	Alias que le asigna un usuario a un nombre. Por defecto será el mismo que el id hasta que el usuario lo cambie
HORA	[HH:MM]
DÍA	[lunes, martes, miércoles, jueves, viernes, sábado, domingo]
FECHA	[DD-MM-YYYY]
TEXTO	Texto del SMS limitado a 160 caracteres.

Al igual que los registros de llamadas dependiendo de la antigüedad del SMS se mostrará un campo u otro.

Ejemplo:

```
<SMS>
  <entrada id="622622622">
          <alias>Bob</alias>
          <hora>11:23</hora>
          <dia>Jueves</dia>
          <fecha>09-05-2013</fecha>
          <texto>Contenido del SMS 1</texto>
  </entrada>
  <entrada id="633633633">
          <alias>Alice</alias>
          <hora>14:51</hora>
          <dia>Viernes</dia>
          <fecha>10-05-2013</fecha>
          <texto>Contenido del SMS 2</texto>
  </entrada>
</SMS>
```

Encuestas contiene los registros con las encuestas realizadas por los usuarios por día. Contiene dos tipos de archivo: base.xml y [dd-mm-yyyy].xml:

Base.xml contiene la raíz del fichero XML. Se utiliza como plantilla cuando no existe un fichero XML para ese día y [dd-mm-yyyy].xml contiene un registro con las encuestas realizadas durante un día en concreto. Sus campos son los siguientes:

Tabla 3 Campos en una entrada de Encuesta

CAMPO	DESCRIPCIÓN
USUARIO	Usuario que realiza la encuesta.
CONTACTO	Contacto con el que nos hemos comunicado.
AUDIO	[-, 1, 2, 3, 4, 5] Valoración de audio en la encuesta.
VIDEO	[-, 1, 2, 3, 4, 5] Valoración del vídeo en la encuesta.
COMENTARIO	Texto con el posible comentario del usuario.
NAVEGADOR	Navegador utilizado.
HORA	Hora a la que se realiza el formulario.

Ejemplo:

```
<ENCUESTA>
  <entrada usuario="622622622">
          <contacto>633633633</contacto>
          <audio>3</audio>
          <video>-</video>
          <comentario>Todo correcto</comentario>
          <navegador>Firefox 21</navegador>
          <hora>19:30</hora>
  </entrada>
  <entrada usuario="633633633">
          <contacto>622622622</contacto>
          <audio>4</audio>
          <video>3</video>
          <comentario>-</comentario>
          <navegador>Chrome 29</navegador>
          <hora>19:00</hora>
  </entrada>
<ENCUESTA>
```

Registro contiene los registros de llamadas de todos los usuarios organizados por días. Contiene dos tipos de archivo: base.xml y [dd-mm-yyyy].xml:

Base.xml contiene la raíz del fichero XML. Se utiliza como plantilla cuando no existe un fichero XML para ese día y [dd-mm-yyyy].xml contiene un registro con las llamadas realizadas durante un día en concreto. Sus campos son los siguientes:

Tabla 4 Campos de una entrada en Registro

CAMPO	DESCRIPCIÓN
USUARIO	Usuario que realiza la encuesta.
CONTACTO	Contacto con el que nos hemos comunicado.
NAVEGADOR	Navegador utilizado.
ESTADO	Indica si la llamada se establece o si es entrante o saliente.
TIEMPO	[HH-MM-SS] Duración de la llamada.
HORA	Hora a la que se realiza el formulario.

Ejemplo:

```
<REGISTRO>
  <entrada usuario="622622622">
          <contacto>633633633</contacto>
          <navegador>Chrome 29</navegador>
          <estado>2</estado>
          <tiempo>0:00:24</tiempo>
          <hora>22:38</hora>
  </entrada>
  <entrada usuario="633633633">
          <contacto>622622622</contacto>
          <navegador>Firefox 25</navegador>
          <estado>4</estado>
          <tiempo>0:00:30</tiempo>
          <hora>16:17</hora>
  </entrada>
<REGISTRO>
```

7.2 Fichero JSON

Para guardar los datos recogidos por la aplicación se ha optado por generar un fichero JSON. Este almacenará los datos suministrados por la aplicación y serán mostrados de manera ordenada en una página web (explicada más adelante).

La decisión de usar en este caso JSON en vez de XML es la simplicidad que nos ofrecen PHP y JavaScript para manejar los datos en este formato.

La estructura del fichero JSON es el siguiente:

```
[
    {
        "fecha": "27-01-2014",
        "hora": "22:45",
        "version": "091213",
        "user": "622622622",
        "contacto": "633633633",
        "estado": "3",
        "duracion": "",
        "navegador": "Chrome 32",
        "usaVideo": "false",
        "audio": "-",
        "video": "-",
        "comentario": "-"
    },
    {
        "fecha": "27-01-2014",
        "hora": "22:45",
        "version": "091213",
        "user": "633633633",
        "contacto": "622622622",
        "estado": "4",
        "duracion": "",
        "navegador": "Chrome 32",
        "usaVideo": "false",
        "audio": "-",
        "video": "-",
        "comentario": "-"
    }
]
```

Contiene todos los campos vistos en los XMLs comentados anteriormente, pero de manera conjunta. De esta manera será posible realizar estadísticas como el número de usuarios que valoran el formulario, que navegador es el más usado o el tiempo total que se ha llamado.

7.3 PHP

En esta sección se explicarán cómo se recogen y se procesan las diferentes peticiones que realiza la aplicación al servidor web.

alias.php es el encargado de cambiar el nombre (definir un alias) para las entradas en la agenda. Este fichero recibe tres parámetros enviados por POST, nombre, oldAlias y newAlias de la siguiente manera:

```
$fileName=$_POST["nombre"];
$oldName=$_POST["oldAlias"];
$newName=$_POST["newAlias"];
```

Ejemplo:

```
newAlias:Alice
nombre:../php/agenda/alice.xml
oldAlias:622622622
```

El código de este fichero obtiene el fichero XML, realiza una búsqueda de todos los nodos con el alias antiguo y lo cambia por el nuevo alias. Al finalizar guarda los cambios.

checkArchivosXML.php comprueba si existe el fichero de agenda y de SMS del usuario creando uno nuevo usando el archivo base.xml en caso de que no existan.

Recibe tres parámetros, el nombre del usuario, el fichero de agenda y el fichero de SMS:

```
$user = $_POST["user"];
$agenda = "../php/agenda/".$user.".xml";
$sms = "../php/sms/".$user."-send.xml";
```

subir.php añade las entradas en la agenda. Estas entradas se hacen creando nodos en ficheros XML, que serán leídos por la aplicación para mostrar su contenido. Este fichero recibe dos parámetros, enviados por POST, contenido, nombre y estado.

```
$fileName=$_POST["nombre"];
$stringXML=$_POST["contenido"];
$state=$_POST["estado"];
```

`nombre` contiene la ruta al fichero XML con el nombre de nuestro usuario. `contenido` indica el nombre que se almacena en el XML, es decir, será el nombre del contacto con el que nos comunicamos y estado indica el estado de la llamada, es decir, si esta se ha establecido o se trata de una llamada entrante o saliente.

Ejemplo:

```
nombre:../php/agenda/alice.xml
contenido:622622622
estado:2
```

El código de este fichero parsea el nombre del usuario (por si lo recibimos como sip:nombre@...) y después comprueba que el fichero XML para dicho usuario ya exista y se creara un nuevo nodo con la información, en el caso de que no exista el fichero XML, se creara uno nuevo leyendo base.xml con la estructura inicial del fichero XML.

Este fichero PHP, contiene una clase para poder añadir usuarios al inicio del XML, para poder mostrar llamadas recientes primero.

sendSMS.php encargado del envió de SMSs. Requiere dos ficheros PHP adicionales kannelConnection.php e insertSMS.php.

Recibe tres parámetros, el número al que se envía el SMS, el cuerpo del mensaje de texto y el nombre del usuario que lo envía:

```
$destinationNumber = $_POST["destinationNumber"];
$textMessage = $_POST["textMessage"];
$user = $_POST["user"];
```

Utiliza la clase kannelConnection.php para el envió del SMS y en caso de respuesta afirmativa lo añade al registro utilizando insertSMS.php.

kannelConnection.php contiene la función `kannelCon($user, $destinationNumber, $textMessage);` que envía una petición al servidor encargado del envió de SMSs. Esta función devuelve una respuesta a la petición que es utilizada por el fichero sendSMS.php para notificar al usuario.

insertSMS.php contiene la función `insertSMS($user, $destinationNumber, $textMessage, $fileName)` para insertar en el fichero del usuario el SMS enviado.

borrar.php utilizado para borrar el registro de llamadas o SMS de un usuario. Recibe un único parámetro con la dirección del fichero que se desea borrar.

polling.php encargado de añadir los datos del cuestionario al fichero XML.

Recibe siete parámetros, la dirección del fichero XML, el nombre de usuario, la valoración de audio, la valoración de vídeo, el comentario del usuario, navegador utilizado y el contacto con el que nos hemos comunicado.

```
$fileName = "../php/encuestas/".date("d-m-Y").".xml";
$user = $_POST["user"];
$audio = $_POST["audio"];
$video = $_POST["video"];
$comments = $_POST["comentario"];
$browser = $_POST["navegador"];
$contacto = $_POST["contacto"];
```

El código de este fichero comprueba si existe un fichero de cuestionarios de ese día cargándolo o creándolo en caso negativo. Posteriormente añade la información al fichero XML y lo guarda.

estadísticos.php guarda en un único fichero JSON todos los datos recogidos por la aplicación para posteriormente poder sacar estadísticos.

Recibe trece parámetros, la fecha actual, la hora, la versión del programa, en nombre del usuario, el contacto, el estado de la llamada, la duración de la llamada, el navegador utilizado, si el usuario ha utilizado vídeo, la valoración de audio, la valoración de vídeo y el comentario en el formulario.

```
$fecha = date("d-m-Y");
$hora = date('H:i');
$version = $_POST["version"];
$user = $_POST["user"];
$contacto = $_POST["contacto"];
$estado=$_POST["estado"];
$duracion=$_POST["duracion"];
$navegador=$_POST["navegador"];

$usaVideo=$_POST["usaVideo"];
$audio=$_POST["audio"];
$video=$_POST["video"];
$comentario=$_POST["comentario"];
```

El código lee el fichero JSON guardado en el servidor con todos los datos, decodifica a un array para añadir un nuevo nodo con los nuevos datos y vuelve a codificarlo en JSON para guardar los cambios.

7.4 Gestión

Se han diseñado cuatro páginas webs adicionales donde visualizar la información recopilada por la aplicación.

Registros (figura 16), nos permite visualizar en una tabla las llamadas realizadas en un día en concreto, mostrando el destinatario de la llamada, la duración, el navegador utilizado y hora a la que se realizó. Además se indicará si la llamada se ha establecido o no, así como si es saliente o entrante mediante flechas verdes, rojas o azules con el sentido de la llamada.

Esta página web escrita en PHP realiza un listado con los archivos XML guardados en el servidor mostrándolos en una lista desplegable. Cuando el usuario selecciona una fecha de las mostradas en la lista anterior se realiza una petición a otro PHP encargado de mostrar la tabla.

Figura 16: Página web "Registros"

Encuestas (figura 17), es la web encargada de mostrar la información recopilada en los cuestionarios. Al igual que en el caso anterior se trata de una web escrita en PHP que realiza un listado con los ficheros XML guardados mostrándolos en una lista desplegable. En este caso cuando el usuario selecciona una opción de la lista se mostrarán las encuestas realizadas en esa fecha indicando las puntuaciones y el comentario dado por el usuario.

Figura 17: Página web "Encuestas"

Panel de usuarios (figura 18)**,** muestra al administrado el fichero de configuración con los usuarios y contraseñas de la aplicación. Esta web está protegida con un usuario y contraseña para que sólo aquellos con autorización puedan modificar las credenciales de los usuarios.

Al igual que las webs anteriores, está escrita en PHP. La web lee el fichero de configuración mostrando su contenido permitiendo al administrador modificarlo y guardar los cambios si así lo desea pulsando el botón situado en la parte inferior.

Estadísticas (figura 19), muestra en una única tabla todos los datos recopilados por la aplicación. Además permite descargar los datos en formato CSV para poder importarlos en una hoja de cálculo y hacer las estadísticas que se necesiten.

Esta web realiza una petición al servidor descargando el fichero JSON y muestra su contenido en una tabla. Debido al gran tamaño que puede alcanzar este fichero, se muestra al usuario una imagen de carga mientras se esté procesando.

Figura 18: Página web "Panel de usuarios"

Fecha	Hora	Version	Usuario	Estado	Contacto	Duracion	Navegador	Usa video	Audio	Video	Comentario
27-01-2014	22:45	091213	820100024	↙	820100025		Chrome 32	false	-	-	-
27-01-2014	22:45	091213	820100025	↙	820100024		Chrome 32	false	-	-	-
27-01-2014	22:45	091213	820100025	↗	820100024	0:00:40	Chrome 32	true	5	5	todo correcto! :)
27-01-2014	22:44	091213	820100024	↙	820100025	0:00:41	Chrome 32	true	-	-	-
27-01-2014	22:26	091213	820100025	↗	622429057	0:00:11	Chrome 32	false	-	-	-
25-01-2014	17:08	091213	820100025	↗	622429057	0:00:13	Chrome 32	false	5	-	todo correcto.
25-01-2014	16:59	091213	820100025	↗	622429057	0:00:37	Chrome 32	false	-	-	-
25-01-2014	14:25	091213	820100025	↗	622429057		Chrome 32	false	-	-	-
23-01-2014	18:32	091213	820100025	↗	622429057	0:00:03	Chrome 32	false	-	-	-

Figura 19: Página web "Estadísticas"

Capítulo 8: Plan de pruebas.

En este capítulo se explicarán las pruebas realizadas durante el desarrollo de la aplicación.

En primer lugar se ha comprobado el correcto funcionamiento del OpenIMSCore. Tras su instalación y añadir los usuarios a la base de datos HSS se procedió a comprobar si el registro y comunicación entre usuarios SIP era correcto. Para ello se utilizó la aplicación gratuita X-lite, un cliente SIP que nos permite conectarnos con nuestro OpenIMSCore indicándole los parámetros adecuados.

Se ha utilizado la aplicación de captura de tráfico Wireshark para monitorizar el intercambio de los paquetes entre la aplicación y el servidor y que el uso de los protocolos empleados sea el correcto.

Como se comentó en capítulos anteriores se dispondrá de una web donde visualizar los registros de las llamadas y los formularios. Estos datos recopilados se han utilizado de pruebas pudiendo registrar posibles fallos analizando el número de llamadas realizadas sin éxito, el navegador utilizado o leyendo los comentarios de los usuarios.

La aplicación ha sido probada en los principales navegadores (Chrome, Firefox e Internet Explorer), comprobando que en todos ellos se visualizaba correctamente la web, y que el funcionamiento era el esperado realizando las siguientes pruebas:

1. Inicio y cierre de sesión.
2. Llamadas entrantes y salientes.
3. Llamada de vídeo.
4. Envió de SMS.
5. Registro del formulario.
6. Registro en la agenda y modificación de alias para un contacto.

8.1 Versión Flash

Durante el desarrollo de la aplicación se han ido probando cada una de las funcionalidades desarrolladas individualmente para garantizar su correcto funcionamiento y una prueba general al finalizar.

Uso de RTMPT para evitar los cortafuegos

Uno de los primeros problemas fue el bloqueo de los mensajes RTMP generados por la aplicación. Algunos firewalls o cortafuegos filtran aquellos paquetes que utilicen un protocolo no tan común como es el caso de RTMP.

Para empezar a utilizar la aplicación el usuario deberá iniciar sesión indicando el usuario y contraseña. La aplicación intentará conectarse al servidor a diferentes direcciones y puertos definidos por el administrador en un fichero XML.

La conexión con el servidor Red5 se realiza utilizando el protocolo RTMP por defecto, por lo que se cambió por RTMPT (RTMP Tunelado). Este protocolo es igual que RTMP pero encapsula los paquetes RTMP en HTML, evitando el bloqueo de los paquetes.

Con Wireshark analizamos el intercambio de los paquetes. En este caso para establecer la conexión primero se envía un mensaje post /fcs/ident2 que recibirá una respuesta de error 404. A continuación se envía un /open/1 y recibirá el identificador que se utilizará en los paquetes siguientes. En el mensaje /send/ se envían los datos Action Message Format (AMF). AMF se utiliza para establecer los parámetros y configuración entre el cliente Flash y el servidor Red5.

Como podemos ver en la siguiente imagen. Los mensajes RTMPT son ahora encapsulados bajo el protocolo HTTP.

Protocol	Info
HTTP	POST /fcs/ident2 HTTP/1.1
HTTP	HTTP/1.1 404 Not Found
HTTP	POST /open/1 HTTP/1.1
HTTP	HTTP/1.1 200 OK
HTTP	POST /idle/XIKBA85T4WD0L/0 HTTP/1.1
HTTP	HTTP/1.1 200 OK
HTTP	POST /send/XIKBA85T4WD0L/1 HTTP/1.1
HTTP	HTTP/1.1 200 OK
HTTP	POST /idle/XIKBA85T4WD0L/2 HTTP/1.1
HTTP	HTTP/1.1 200 OK
HTTP	POST /idle/XIKBA85T4WD0L/3 HTTP/1.1
HTTP	HTTP/1.1 200 OK
HTTP	POST /idle/XIKBA85T4WD0L/4 HTTP/1.1

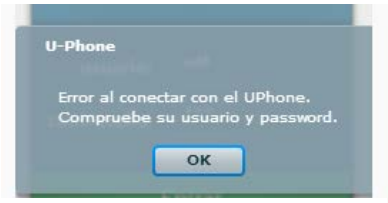
Figura 20: Captura en Wireshark de paquetes RTMPT

Registros generados por el servidor Red5.

El servidor Red5 genera diferentes registros donde poder monitorizar la comunicación entre la aplicación Flash y el servidor y este con el OpenIMSCore. Guardando los mensajes SIP y los comandos RTMP.

Mensajes Alert en ActionScript.

ActionScript permite generar mensajes emergentes con el texto que deseemos, muy útil si queremos mostrar alguna notificación al usuario o mostrar en estos mensajes los valores de las variables que queramos comprobar.



Un ejemplo:

```
Alert.show("Texto a mostrar", "Título");
```

Figura 21: Ejemplo mensaje Alert

8.2 Versión HTML5.

En la versión HTML5 se ha empleado principalmente las herramientas de desarrollador proporcionadas por el navegador. Estas herramientas nos permiten visualizar el código HTML, CSS y JavaScript de la web, el intercambio de lo los archivos descargados y las peticiones realizadas al servidor. Además de poder editar el código localmente permite ejecutar funciones JavaScript y visualizar los registros así como poder ver los datos guardados por la aplicación.

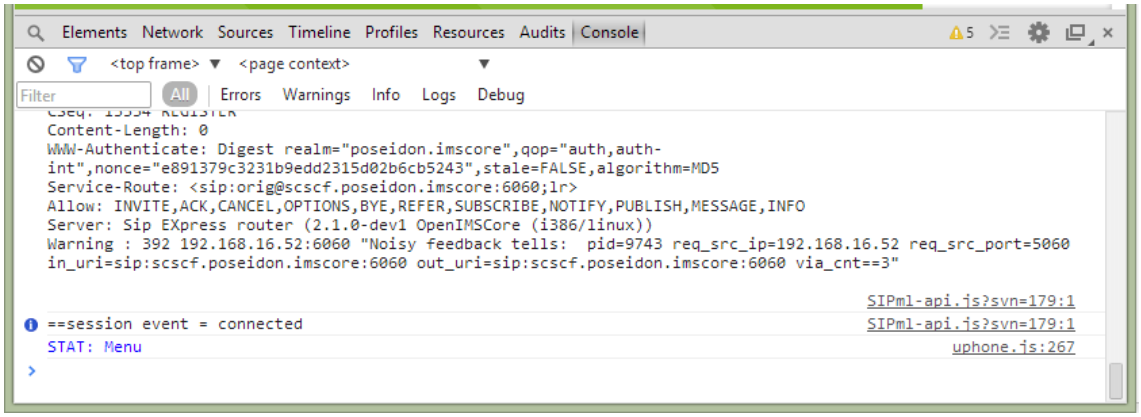


Figura 22: Consola del navegador Chrome

En la figura 22, se muestra la consola del navegador. En esta consola se muestran los mensajes generados por nuestra aplicación o en azul código para depurar.

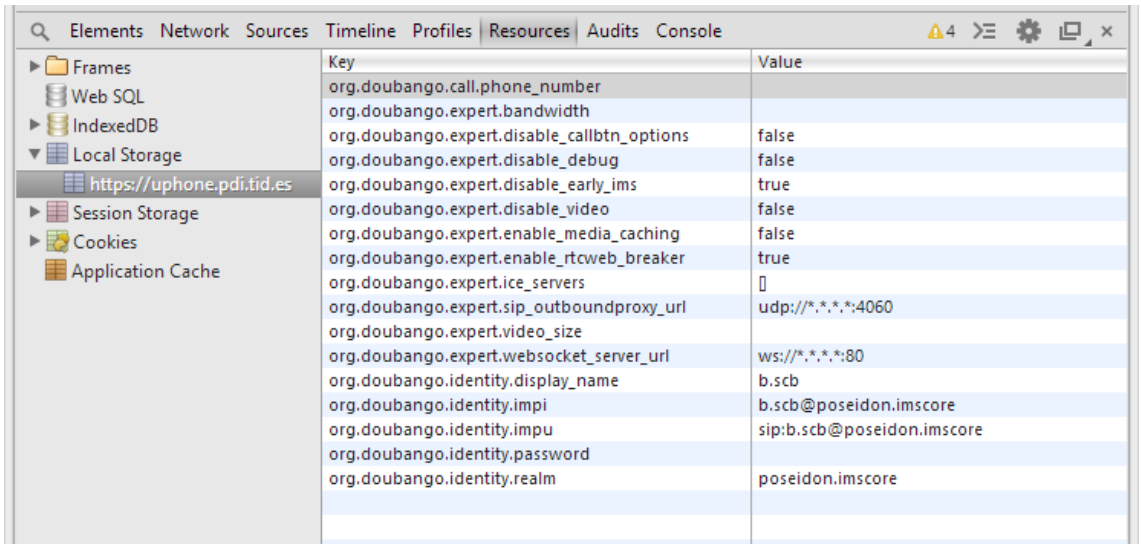


Figura 23: Herramientas de desarrollador de Chrome. Recursos

En la figura 23 se muestra el uso de las herramientas del navegador y cómo podemos visualizar los datos almacenados por la aplicación e incluso modificarlos.

8.3 Versión Móvil.

La versión móvil requiere la utilización de un dispositivo para realizar las pruebas. Durante el desarrollo se ha utilizado el móvil Nexus 4 de Google con sistema operativo Android 4.4 y utilizando el navegador Chrome.

Las **herramientas de desarrollador de Android**, mediante su SDK, permiten la comunicación del navegador del dispositivo con el navegador del PC. De esta manera conectando el dispositivo móvil por USB al PC es posible realizar modificaciones en el código, ejecutar código JavaScript o acceder a la consola del navegador del dispositivo directamente desde el PC.

En la siguiente figura se muestra en el PC la consola del navegador Chrome del móvil. A la izquierda se muestra una simulación de lo que el usuario ve en terminal móvil y a la derecha la consola del navegador del terminal móvil.

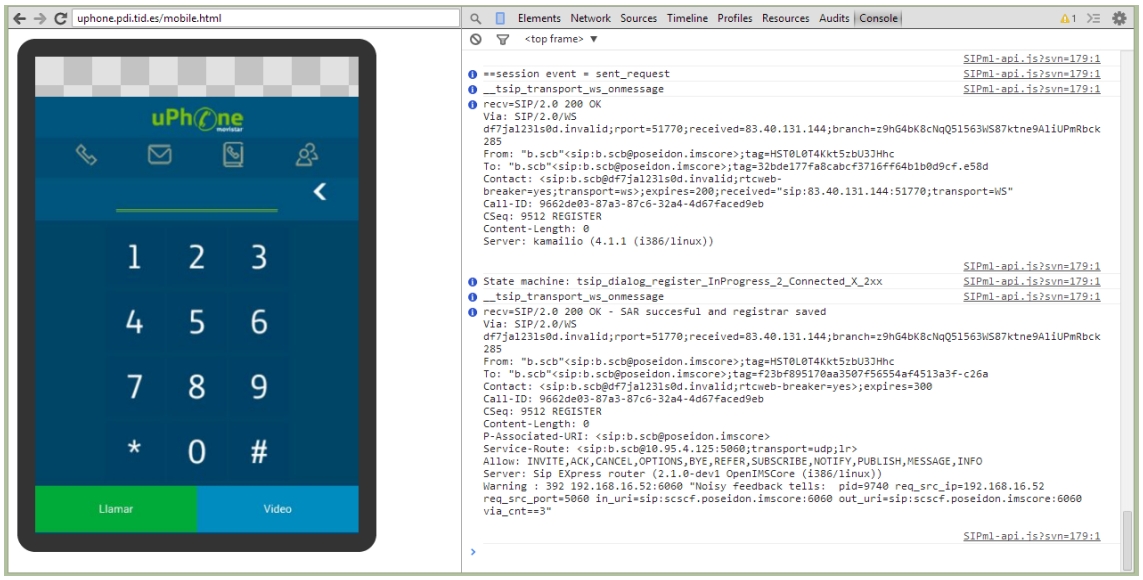


Figura 24: Herramientas de desarrollador en terminal móvil

Capítulo 9: Conclusiones y trabajos futuros.

Como conclusión se puede destacar que la aplicación Flash funciona según lo previsto ofreciendo todas las características. Pero en la actualizad con la aparición de nuevas tecnologías se prevé que Flash desaparezca, dando paso a una web de aplicaciones puramente escritas en HTML5 que no requieran la instalación de ningún componente adicional y sea compatible con cualquier dispositivo.

Como hemos visto, la aplicación desarrollada en Flash únicamente estaba disponible para escritorio, mientras que con HTML5 se podía utilizar tanto en PC, móvil o tabletas. Tecnologías como WebRTC suponen un gran avance con unas grandes funcionalidades pero el aún desarrollo y estandarización de éstas supone cierta inestabilidad además de no ofrecer todas las funcionalidades en todos los navegadores.

Por otro lado, la aplicación desarrollada ha sido pensada para un número limitado de usuarios. La implementación de guardar los registros de las llamadas o los formularios en ficheros XML no serían convenientes si se tratase de un elevado número de usuarios utilizando la aplicación al mismo tiempo.

Como trabajos futuros relacionados con la aplicación, se propone avanzar con el desarrollo de la aplicación HTML5 y cambiar los registros mediante escritura en ficheros XML por bases de datos SQL.

Por otro lado, aprovechando la continua evolución de los nuevos APIs se pueden añadir nuevas funcionalidades y perfeccionar las ya existentes como por ejemplo notificaciones en dispositivos móviles o utilizar WebRTC para transferencia de ficheros además de audio y vídeo.

Finalmente, otro trabajo futuro sería el desarrollo de una aplicación móvil que añadiese nuevas funcionalidades como acceso a los contactos.

Presupuesto.

A continuación se presenta una estimación detallada del presupuesto necesario para el desarrollo de este trabajo.

Los costes se presentan en tres grupos, Software, Equipos y personal. No se incluyen costes de dietas o viajes y el tiempo de amortización estimado para los equipos será de 5 años y 3 para el de Software. La tasa de costes indirectos será del 20% y un IVA del 21%. El coste de personal incluye los costes para la empresa como el alta en la seguridad social.

Tabla 5 Costes de Software

DESCRIPCIÓN	COSTE	DEDICACIÓN	COSTE IMPUTABLE
ADOBE PHOTOSHOP CC	12€/mes	9 meses	108,00€
ADOBE FLASH BUILDER 4.6	180€	9 meses / 3 años = 25%	45,00€
OFFICE HOGAR Y EMPRESAS 2013	269€	9 meses / 3 años = 25%	67,25€
VISIO PROFESSIONAL 2013	739€	9 meses / 3 años = 25%	184,75€
PROJECT STANDARD 2013	769€	9 meses / 3 años = 25%	192,25€
APACHE HTTP	0€	9 meses	0,00€
OPEN IMSCORE	0€	9 meses	0,00€
RED5SERVER	0€	9 meses	0,00€
RED5PHONE	0€	9 meses	0,00€
WEBRTC2SIP	0€	9 meses	0,00€
SIPML5	0€	9 meses	0,00€
TOTAL			**597,25€**

Tabla 6 Costes de Equipos

DESCRIPCIÓN	COSTE	DEDICACIÓN	COSTE IMPUTABLE
PC DE SOBREMESA	600€	9 meses / 5 años = 15%	90,00€
EQUIPO IMS	400€	9 meses / 5 años = 15%	60,00€
EQUIPO APACHE +RED5 +WEBRTC2SIP	400€	9 meses / 5 años = 15%	60,00€
EQUIPO KANNEL	300€	9 meses / 5 años = 15%	45,00€
GATEWAY VOIP CISCO SPA3102	35€	9 meses / 5 años = 15%	5,25€
SBC ACME PACKET 4500	13.160€	9 meses / 5 años = 15%	1.974,00€
TOTAL			**2.234,25€**

Tabla 7 Costes de personal

DESCRIPCIÓN	COSTE	DEDICACIÓN	COSTE IMPUTABLE
INGENIERO SENIOR	100€/hora	20h/mes durante 9 meses	18.000€
INGENIERO JUNIOR	50€/hora	80h/mes durante 9 meses	36.000€
TOTAL			**54.000€**

Tabla 8 Costes totales

DESCRIPCIÓN	COSTE IMPUTABLE	IVA (21%)	COSTE FINAL
SOFTWARE	597,25€	125,42€	722,67€
EQUIPOS	2.234,25€	469,19€	2.703,44€
PERSONAL	54.000,00€	11.340,00€	65.340,00€
COSTES INDIRECTOS (20%)	11.366,30€	2.386,92€	13.753,22€
TOTAL			**82.519,34€**

Marco regulador

La aplicación desarrollada se publica en el servicio Zona Beta de Movistar, una plataforma donde los usuarios pueden probar los servicios y productos expuestos. Por consiguiente las condiciones de uso y utilización de la aplicación están basadas en los del servicio Zona Beta [31].

Se quiere destacar del enlace anterior el apartado de Protección de datos de carácter personal (LOPD). La aplicación desarrollada en este trabajo no guarda información del usuario que permita identificarle directamente, sino que únicamente hace un registro de datos con fines estadísticos, lo que queda recogido en las condiciones que el usuario deberá aceptar para acceder a la Zona Beta. Cito textualmente:

> *"…el Usuario consiente expresamente que, al participar en los productos precomerciales, Telefónica de España y Telefónica Móviles traten los datos de uso, tráfico y navegación de dichos servicios con fines estadísticos, datos que se tratarán de forma disociada."*

Por otro lado, estos datos recopilados son utilizados con el propósito de proporcionar el servicio a los usuarios lo cual está permitido por la LOPD.

Otro punto importante a tener en cuenta, sería la ley 25/2007, de 18 de octubre, de conservación de datos relativos a las comunicaciones electrónicas y a las redes públicas de comunicaciones.

> *"…la obligación de los operadores de telecomunicaciones de retener determinados datos generados o tratados por los mismos, con el fin de posibilitar que dispongan de ellos los agentes facultados.*
>
> *…éstos puedan obtener los datos relativos a las comunicaciones que, relacionadas con una investigación, se hayan podido efectuar por medio de la telefonía fija o móvil, así como por Internet…*
>
> *…los datos sobre los que se establece la obligación de conservación son datos exclusivamente vinculados a la comunicación, ya sea telefónica o efectuada a través de Internet, pero en ningún caso reveladores del contenido de ésta; y, en segundo lugar, que la cesión de tales datos que afecten a una comunicación o comunicaciones concretas, exigirá, siempre, la autorización judicial previa."*

Aunque al trabajo presentado supone un sistema en pruebas y no da soporte por el momento a dicha ley, deberá darlo si se implanta en producción.

Entorno socio-económico.

Según la Comisión de Mercado de las Telecomunicaciones (CMT), Los ingresos del 2012 en telefonía fija cayeron un 10,7%. El número de líneas y el tráfico generado continuaron descendiendo y los operadores alternativos continuaron ganando terreno alcanzando un 23,3% de las líneas.

Los ingresos en banda ancha fija sufrieron una caída del 4,6% aunque el número de líneas aumentó un 3,2% y las líneas con una velocidad de conexión de 30Mbps o superior crecieron un 63,7%. El precio medio de las ofertas de banda ancha y voz disminuyeron al igual que el gasto en el hogar y los precios de las llamadas. [32]

Por otro lado, los ingresos en la telefonía móvil cayeron en 2012 un 15,9%, siendo un 3,7% menos el número de líneas móviles contratadas. Por el contrario los ingresos por la banda ancha móvil sufrieron un crecimiento del 29% siendo un 44,2% el incremento de suscripciones. [33]

Tabla 9 Ingresos del tercer cuatrimestre en los últimos años.

Ingresos totales (millones de euros)			
Trimestre/Año	**III/2011**	**III/2012**	**III/2013**
Telefonía fija	1.316,13	1.164,08	1.009,71
Banda ancha	940,48	894,40	869,39
Telefonía móvil	2.900,24	2.403,04	1.892,18
Banda ancha móvil	595,14	713,22	853,59

En la tabla superior podemos ver los resultados de los ingresos en el tercer trimestre de los últimos tres años. Como se puede observar los ingresos disminuyen año a año excepto en la banda ancha móvil. [34]

Las cifras anteriores ponen a las operadoras en la situación de buscar nuevas soluciones y ofrecer nuevos servicios que les permitan aumentar su cuota de mercado. Este proyecto se enmarca en esta línea ofreciendo un servicio complementario a la línea de teléfono convencional.

Lista de figuras

FIGURA 1: DIAGRAMA DE GANTT 4
FIGURA 2: COMUNICACIÓN CLIENTE-SERVIDOR UTILIZANDO EL PROTOCOLO RTMP 9
FIGURA 3: INTERFAZ APLICACIÓN RED5PHONE 12
FIGURA 4: SOPORTE DE WEBRTC SEGÚN NAVEGADOR. 16
FIGURA 5: TOPOLOGÍA DE COMUNICACIÓN WEBRTC ENTRE DOS NAVEGADORES 17
FIGURA 6: TOPOLOGÍA DE COMUNICACIÓN ENTRE NAVEGADOR RED IMS/PSTN 17
FIGURA 7: API WEBRTC - FUENTE: WEBRTC.ORG 19
FIGURA 8: ARQUITECTURA WEBRTC2SIP - FUENTE: WEBRTC2SIP.ORG 21
FIGURA 9: SIPML5 LIVE DEMO. 21
FIGURA 10: TOPOLOGÍA APLICACIÓN FLASH 25
FIGURA 11: TOPOLOGÍA APLICACIÓN HTML5 25
FIGURA 12: DISEÑO APLICACIÓN FLASH 26
FIGURA 13: DIAGRAMA DE ESTADOS 28
FIGURA 14: APLICACIÓN HTML5 EN ESCRITORIO 35
FIGURA 15: APLICACIÓN HTML5 EN MÓVIL 39
FIGURA 16: PÁGINA WEB "REGISTROS" 49
FIGURA 17: PÁGINA WEB "ENCUESTAS" 50
FIGURA 18: PÁGINA WEB "PANEL DE USUARIOS" 51
FIGURA 19: PÁGINA WEB "ESTADÍSTICAS" 51
FIGURA 20: CAPTURA EN WIRESHARK DE PAQUETES RTMPT 53
FIGURA 21: EJEMPLO MENSAJE ALERT 53
FIGURA 22: CONSOLA DEL NAVEGADOR CHROME 54
FIGURA 23: HERRAMIENTAS DE DESARROLLADOR DE CHROME. RECURSOS 54
FIGURA 24: HERRAMIENTAS DE DESARROLLADOR EN TERMINAL MÓVIL 55

Lista de tablas

TABLA 1 CAMPOS DE UNA ENTRADA EN LA AGENDA 41
TABLA 2 CAMPOS DE UNA ENTRADA EN SMS 42
TABLA 3 CAMPOS EN UNA ENTRADA DE ENCUESTA 43
TABLA 4 CAMPOS DE UNA ENTRADA EN REGISTRO 44
TABLA 5 COSTES DE SOFTWARE 57
TABLA 6 COSTES DE EQUIPOS 58
TABLA 7 COSTES DE PERSONAL 58
TABLA 8 COSTES TOTALES 59
TABLA 9 INGRESOS DEL TERCER CUATRIMESTRE EN LOS ÚLTIMOS AÑOS 61

Acrónimos

A

AMF
Action Message Format 53
API
Application Programming Interface 15, 18, 19, 21, 29, 40

C

CMT
Comisión de Mercado de las Telecomunicaciones 61
CSCF
Call Session Control Function 11
CSS
Cascading Style Sheets 14, 15, 35, 54

D

DoS
Denial of Service 24

F

FCS
Flash Communication Server 8
FMS
Flash Media Server 8

H

HSS
Home Subscriber Server 11, 52
HTML
HyperText Markup Language 5, 14, 23, 35, 40, 53, 54
HTTPS
Hypertext Transfer Protocol Secure 17, 24

I

ICE
Interactive Connectivity Establishment 18, 20
IETF
Internet Engineering Task Force 16
IMS
IP Multimedia Subsystem 10
IP
Internet Protocol 5, 6, 8, 10, 14, 18, 25

J

JSON

JavaScript Object Notation 41, 45, 48, 50

L

LOPD
Ley Orgánica de Protección de Datos 60

M

MXML
Macromedia eXtensible Markup Language 13, 33

N

NAT
Network Address Translation 18, 25

P

PHP
Hypertext Pre-processor 2, 37, 41, 45, 46, 47, 49, 50
PSTN
Public switched telephone network 20, 24

Q

QoS
Quality of Service 24

R

REST
Representational State Transfer 18
RIA
Rich Internet Applications 12
RTMP
Real Time Messaging Protocol 2, 5, 8, 9, 10, 12, 13, 14, 24, 29, 32, 52, 53
RTP
Real-time Transport Protocol 25

S

SBC
Session Border Controler 24, 25
SCTP
Stream Control Transmission Protocol 18
SDK
Software Development Kit 55
SIP
Session Initiation Protocol ..... i, iii, v, 2, 5, 6, 7, 11, 14, 17, 18, 20, 21, 24, 25, 32, 41, 42, 52, 53
SMPP
Short Message Peer-to-Peer 24
SMS
Short Message Service ..... v, 2, 24, 27, 42, 43, 46, 47, 52
SMSC

Short Message Service Center 24
SQL
Structured Query Language 56
SSL
Secure Sockets Layer 8
STUN
Session Traversal Utilities for NAT 18, 25
SVG
Scalable Vector Graphics 15
SWF
Small Web Format 11, 12

T

TSL
Transport Layer Security 8
TURN
Traversal Using Relay NAT 18

W

W3C
World Wide Web Consortium 14, 16
WHATWG
Web Hypertext Application Technology Working Group 14
WOFF
Web Open Font Format 15

X

XML
Extensible Markup Language 2, 14, 33, 41, 42, 43, 44, 45, 46, 47, 48, 49, 52, 56
XMPP
Extensible Messaging and Presence Protocol 18

Referencias

[1] R. Estepa, «Evolución histórica de las telecomunicaciones,» Diciembre 2004. [En línea]. Available: http://trajano.us.es/~rafa/ARSS/apuntes/tema1.pdf. [Último acceso: 2014].

[2] J. F.Kurose, «Redes de computadoras. Un enfoque descendente,» Pearson, 2010, pp. 607 - 613.

[3] G. Camarillo, «SIP desmystified,» McGraw-Hill, 2001.

[4] H. S. J. Rosenberg, «Reliability of Provisional Responses in the Session Initiation Protocol (SIP). RFC 3252. Internet Engineering Task Force,» Junio 2002. [En línea].

[5] J. Rosenberg, «A Hitchhiker's Guide to the Session Initiation Protocol (SIP). RFC 5411. Internet Engineering Task Force,» Enero 2009. [En línea].

[6] R. Reinhardt, «Adobe® Flash® CS3 Professional Video Studio Techniques,» pp. 65 - 71.

[7] Adobe, «Adobe's Real Time Messaging Protocol,» 21 Diciembre 2012. [En línea]. Available: http://www.adobe.com/content/dam/Adobe/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf.

[8] M. Poikselkä, The IMS: IP Multimedia Concepts and Services, 2nd Edition, Wiley, 2007.

[9] G. Camarillo, The 3G IP Multimedia Subsystem (IMS), Wiley, 2008.

[10] «OpenIMSCore.org | The Open Source IMS Core Project,» [En línea]. Available: http://www.openimscore.org/.

[11] A. Systems, ADOBE® FLEX® 3: PROGRAMMING ACTIONSCRIPT™ 3.0, Adobe Systems, 2008.

[12] «Apache Flex,» [En línea]. Available: http://flex.apache.org/.

[13] «Red5 - The Open Source Media Server,» [En línea]. Available: http://www.red5.org/.

[14] «red5phone - An open source SIP Phone for Adobe Flash using Red5,» [En línea]. Available: https://code.google.com/p/red5phone/.

[15] «HTML5 Rocks - Un recurso para desarrolladores de HTML5 para una Web abierta,» [En línea]. Available: http://www.html5rocks.com/es/.

[16] «W3Schools Online Web Tutorials,» [En línea]. Available: http://www.w3schools.com/.

[17] «HTML5,» [En línea]. Available: http://www.w3.org/TR/html5/.

[18] J. Rosenberg, «Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245. Internet Engineering Task Force,» Abril 2010. [En línea].

[19] R. M. P. M. D. W. J. Rosenberg, «Session Traversal Utilities for NAT (STUN). RFC 5389. Internet Engineering Task Force,» Octubre 2008. [En línea].

[20] P. M. J. R. R. Mahy, «Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766. Internet Engineering Task Force.,» Abril 2010. [En línea].

[21] S. Dutton, «Getting Started with WebRTC - HTML5 Rocks,» 2012. [En línea]. Available: http://www.html5rocks.com/en/tutorials/webrtc/basics/. [Último acceso: 2014].

[22] «API de MediaStream - WebRTC | MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/WebRTC/MediaStream_API.

[23] «Comunicaciones peer-to-peer (P2P) con WebRTC - WebRTC | MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/WebRTC/Peer-to-peer_communications_with_WebRTC.

[24] «webrtc2sip - Smart SIP and Media Gateway to connect WebRTC endpoints,» [En línea]. Available: http://webrtc2sip.org/.

[25] «sipML5 - The world's first open source HTML5 SIP client,» [En línea]. Available: http://sipml5.org/.

[26] «NetConnection - Adobe ActionScript 3 API Reference,» [En línea]. Available: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/NetConnection.html.

[27] «NetStream - Adobe ActionScript 3 API Reference,» [En línea]. Available: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/NetStream.html.

[28] «HTML DOM Event Object,» [En línea]. Available: http://www.w3schools.com/jsref/dom_obj_event.asp.

[29] S. M. M. B. A. B. Doug Schepers, «Touch Events. W3C,» Octubre 2013. [En línea]. Available: http://www.w3.org/TR/touch-events/.

[30] A. Kostiainen, «Vibration API. W3C,» Febrero 2014. [En línea]. Available: http://www.w3.org/TR/vibration/.

[31] «Condiciones de uso del servicio Zona Beta,» [En línea]. Available: https://zonabeta.movistar.es/condiciones-de-uso.

[32] «Comunicaciones fijas - Informe 2011 - CMT,» [En línea]. Available: http://informecmt.cmt.es/informe-economico-sectorial/comunicaciones-fijas.

[33] «Comunicaciones móviles - Informe 2011 - CMT,» [En línea]. Available: http://informecmt.cmt.es/informe-economico-sectorial/comunicaciones-moviles.

[34] «CNMCData - Comision Nacional de los Mercados y la Competencia,» [En línea]. Available: http://cmtdata.cmt.es/cmtdata/.